Towards Fit for Purpose Security in High Assurance and Agile Environments

Linda Finch

Thesis submitted to Cardiff University for the degree of Doctor of Philosophy

School of Engineering
Communications Research Centre
Cardiff University

January 2009

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Thesis Summary

This thesis examines the problems that arise when military organisations and communities seek to exploit the opportunities provided by Network Enabled Capability for resource sharing and collaboration. The research shows that current methods of assessing security do not take account of the temporal nature of real communications with ad hoc and policy driven connections. It is contended that this results in an overly conservative approach, and that explicit consideration (and implementation) of non-persistent and alternative connectivity would provide better assessments on which to base risk decisions for 'fit for purpose' security, enabling more flexible and responsive security in rapidly changing environmental conditions.

List Of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

Abbreviation	Definition
AODV	Ad-Hoc Distance Vector
AG	Attack Group
AVC	Access Vector Cache
BLP	Bell and LaPadula (security policy model)
CCRA	Common Criteria Recognition Arrangement
CESG	Communications-Electronics Security Group
COTS	Commercial Off The Shelf
CRAMM	Central Computer and Telecommunications Agency (CCTA) Risk Analysis Methodology
DBSy®[1]	Domain Based Security Architecture
DPEA	Dynamic Policy Enforcement Agent
EAL	Evaluation Assurance Level
EM	Environment Monitor
EvM	Event Manager
FOI	Focus of Interest
GPS	Global Positioning System
GUI	Graphical User Interface
HMG	Her Majesty's Government
InfoSec	Information Security
IP	Internet Protocol
IS1	InfoSec Standard number 1
ITL	Interval Temporal Logic
ITSEC	Information Technology Security Evaluation and Certification
MILS	Multiple Independent Levels of Security
MLS	Multi-Level Security
NEC	Network Enabled Capability
NPC2	Non-Persistent Capability Concept
NPD	Non-Persistent Domain

[1] DBSy is a Registered Trademark of QinetiQ.

NSA	National Security Association
RAdAC	Risk Adaptive Access Control
RBAC	Role Based Access Control
RTOS	Real Time Operating System
SANTA	Security Analysis Toolkit for Agents
SELinux	Security Enhanced Linux
SIPC	Secure Inter-Process Communications
SLIDE	SELinux Integrated Development Environment
SoS	System of Systems
SPAT	Security Policy Analysis Toolkit
STRL	Software Technology Research Laboratory
UDP	User Datagram Protocol
2IC	Second in Command

Acknowledgements

Many individuals provided support during this research activity. However, the author would like to particularly acknowledge and thank the following individuals for their assistance at various points in the production of this thesis.

- Dr. Helge Janicke of DeMontfort University for his collaboration on the formal policy modelling work in Chapter 3, Section 3.4.
- Dr. Antonio Cau and Professor Hussein Zedan of DeMontfort University for supporting this collaborative investigation.
- Kay Hughes of QinetiQ for her helpful comments during early development of the Non-Persistent Capability Concept and for donating the material in Appendix A.
- Richard Vaughan for his software development skills and inspired naming of the NPC2 software applications in Chapter 6.
- Dr. Andrew Tilbrook of General Dynamics United Kingdom Limited for instigating and supporting the research activity.
- My supervisor Professor Ken Lever from the Communications Research Centre at Cardiff University for his patience and inspiration during all stages of the research.
- My husband, Stephen Fisher for reading and commenting on numerous papers and sharing the ups and downs of the study.

The author acknowledges the support of her employer, General Dynamics United Kingdom Limited, who sponsored this research programme.

The author has applied the Domain Based Security (DBSy®) approach extensively during this research and acknowledges that DBSy is a registered trademark of QinetiQ.

Thesis Abstract

This thesis examines the problems that arise when military organisations and communities seek to exploit the opportunities provided by Network Enabled Capability for resource sharing and collaboration. A key advantage of Network Enabled Capability is increased flexibility and adaptability to changing resource needs. However, the traditional approach to security, which is suited to static operational regimes, is totally out of step with the agile and flexible requirements of Network Enabled Capability. In this regime, a different approach to engineering security solutions is necessary, which we term *fit for purpose security*. This recognises that in contemporary military operational environments, security must adapt and respond to the changing needs of an organisation but maintain cognisance of appropriately high levels of assurance.

The main contribution of this thesis is the development of a methodology for providing the necessary flexibility in security. The methodology is a synthesis of security *policy*, security *architecture* and security *controls*. To implement flexible security, a new concept is introduced: the Non-Persistent Capability Concept (abbreviated to NPC2), which models the use of conditional network connectivity, installed to provide flexibility. The NPC2 can be regarded as an 'intelligent' connector, capable of detecting and responding to changes in the environment which necessitate a modification to the security policy and user capabilities in the system. The thesis provides examples in which the NPC2 is used to enhance graphical state-of-the-art models used for system planning – which are very useful for static operational scenarios based on worst-case maximum permitted connectivity – but less so for dynamic scenarios requiring adaptivity, where worst case scenarios may never materialise. A fundamental part of the fit for purpose security vision is the provision of techniques for reasoning about the level of potential risk that flexibility introduces into the system. It is shown that the NPC2 provides this ability: examples are given showing how the NPC2 can be incorporated into the process of risk analysis to calculate the effects of modifying the separation and sharing profile of a system. In an attempt to employ analytical techniques that are formalisable – and therefore potentially certifiable – connectivity matrices are used to model the way sharing can take place between the parts of a system. The entries in the matrices are not restricted in the conventional way to '1' or '0' (representing the presence or absence of connectivity), but can take Boolean values (representing the conditional nature of NPC2 connections), or numerical probability values (representing the reliability of particular communication links), or numerical penalty values (reflecting the risks associated with particular sharing arrangements) – or all values simultaneously. The result of mixing real and Boolean arithmetic in a single matrix is shown to enhance the view of the potential and actual risk to a system under dynamically changing conditions. In effect, *total* vulnerability can be decomposed into two elements: *inevitable* and *avoidable,* the former representing the *minimum* vulnerability and the latter representing the *maximum* vulnerability which can be controlled using the NPC2.

In addition to demonstrating the ways in which the NPC2 can provide the conceptual basis for flexible security, the thesis also examines the ways in which the concept can be implemented in practice – but within a technology independent framework, so as to allow a wide variety of hardware and software design options. A small-scale system demonstrating the NCP2 architecture hosted on SELinux has been implemented and successfully tested in a realistic environment; and the results of the experiment are presented.

The research is also taken a stage further by the introduction of another concept, the Non-Persistent Domain (NPD), aimed at modelling ad-hoc collaboration activities and their impact on potential risk. Although it has not yet been carried through to implementation, the NPD is shown to be useful in high-level modelling and risk analysis using the same test environment as the NPC2.

The conclusion of this research is that two non-persistent constructs, the Non-Persistent Capability Concept and the Non-Persistent Domain, provide an improvement in modelling ability needed to move away from static security solutions appropriate to fixed operational scenarios towards the flexible fit for purpose security solutions suited to Network Enabled Capability – whilst still retaining the ability to control the levels of potentially increased risk that flexibility entails. Together, the NPC2 and the NPD provide a means for specifying, designing and analysing flexible security; and also a means for implementing it.

Chapter 1: Introduction

The Information Age [1] is characterised by an increasing requirement to share resources between communities of users with common interests and objectives. These communities may contain members of the same organisation and/or collaborative partners depending on need. Collaboration may be ad-hoc in nature where groups are formed and dispersed spontaneously and have a rapidly changing membership as conditions in the environment dictate. Alternatively the groups may evolve over time and have a greater degree of permanence in terms of the partners involved and nature of the alliance. In either case the resources that are shared between the communities will probably be dispersed across numerous distributed data repositories and accessed from a variety of logical and physical locations using a range of different electronic devices. This business model undoubtedly enables unprecedented opportunities for the rapid collection and dissemination of information and collaboration which in turn can be used to competitive advantage by the organisations involved. However, opportunity has a price and in this case one notable disadvantage arises from balancing the security of valuable assets against the benefits of sharing.

In order to maintain security principles such as confidentiality and integrity boundaries between users must be defined. These identify who (or what) may share with whom and 'from whom (or what) information must also be separated. The Information Age has extended and blurred these traditional boundaries of the network. In a dynamic environment the perimeter of the network, which separates trusted and un-trusted communities of users, is not always obvious. The term 'de-perimeterisation' has been coined by the Jericho forum [2] to describe this phenomenon. In this situation, it clearly becomes more difficult to locate appropriate security controls to provide adequate protection whilst permitting the capabilities required for the organisation to share its resources. Information is regarded as a valuable asset and its compromise can have devastating effects on both the owning organisation and other communities. It is therefore absolutely fundamental to understand and adequately balance the need for sharing against the need to separate in a changing environment.

The desire to participate in electronic collaboration and information sharing is not unique to any one particular type of business domain. The defence community, and more specifically the military, are affected by the Information Age as much as any commercial enterprise [3], although their motivation for participating in collaborative relationships naturally differs.

In the United Kingdom, the military are evolving towards 'Network Enabled Capability (NEC)' where the objective is to enhance military capability through improved exploitation of information [4]. In this respect NEC exhibits similar tenets to that of the Information Age. The NEC vision is to collect, interpret and share information from different sources and system platforms, facilitated by the network. Within the overall programme to realise this vision a number of different themes have been identified, some of which are of particular relevance to this research. These include the requirement for flexible working and reconfiguration of systems to meet changing needs, support of agile mission groups including dynamic creation and configuration of mission groups to share information and a resilient information infrastructure which is managed, secured and allows secure access but also provides flexibility to meet the needs of the agile mission group.

Traditionally the military has operated on a strict separation and sharing policy for resources where a protective marking is applied to identify the potential sensitivity or importance of the resource and separate it from other categories. The ability to share is controlled using an access control policy where the security context of the requesting user is authorised for a set of protective markings. Of course, higher levels of sensitivity or protective marking require stronger assurance that separation and sharing can be and is indeed maintained in order to prevent the loss of confidentiality and integrity. Typically, protective markings and security clearance change very rarely. During the design of a system it is necessary to consider the extremes of protective markings and clearances which effectively means that the designer is always working with the 'worst case' scenario in order to cover all possible cases. This approach can lead to over-classification of some resources and potentially impede access to legitimate users if errors in the design or the labelling of resources have occurred.

With the advent of NEC, a more flexible approach to information security is required where deployed systems are architected to deal with adaptations to the separation and sharing profile as a result of changing conditions in the environment. For example, an agile group may have to collaborate with different and possibly unexpected communities as an operation unfolds. Levels of trust between communities may alter thus affecting the resources that are both shared and separated. The tempo of operations is such that any change to the separation and sharing infrastructure must occur in a timely and assured manner where, as far as possible, the ramifications of introducing change are fully understood and articulated. It is suggested that the degree of flexibility required by the military organisation of the future may be at odds with the axioms of high assurance. Furthermore, unless this imbalance is overcome it is postulated that for the military, information security may be the most significant risk to successful realisation of the network-enabled organisation.

In essence, the Law of Requisite Variety [5] epitomises our research. Simply put, Ashby's law proposes that a flexible system with many options is better able to cope with change. There will of course be a level of compromise. Ashby recognised that a system tightly optimised for an initial set of conditions may be more efficient, whilst those conditions prevail. However, when change occurs the system is likely to fail since it does not have the capacity to respond to a different state in an appropriate manner. Although this law was applied to the development of control systems, it is believed that the principles are also of merit within the field of information security where the construction of more robust and adaptable solutions is required. Although different access control rules may be specified in the security policy to allow changes in capability, a flexible solution also requires the capacity to *select* and *enforce* these different rules or sets of rules when an appropriate change in conditions is detected. It is postulated that such an approach would provide security suited to the situation at the time it is required.

Flexibility is derived, at least in part, from the ability to modify separation and sharing capabilities in accordance with changing conditions in the environment. The research described in this thesis indicates that current technologies and approaches are not adequate to accommodate the level of flexibility required for both immediate and future modes of operation. System design and development are heavily biased

towards security through separation with no allowance made for temporary deviations from the model. It is the opinion of the author that this approach impedes flexibility and resource sharing. Of course expediency must be tempered by constraints; however this does not change our view that a more radical approach to the challenge is required. In response this work builds upon the concept of Ashby's law to propose the creation of *fit for purpose* security within the military. This approach adopts a more realistic view of the system by considering fluctuations in capability (and therefore separation and sharing requirements) and thus the option to architect and enforce different controls at appropriate points in the system.

The concept of fit for purpose security has been applied, albeit differently from our interpretation, in work from Digital Rights Management. In [6, p.180] Kuhlmann and Ghering suggest that:

"...in everyday situations, security is a flexible notion rather than an absolute goal. In order to be trustworthy a system has to be sufficiently secure to be fit for purpose."

The above quotation was part of a general discussion on the role of trusted platform technology [7] and in particular, its potential use in Digital Rights Management. Kuhlmann and Gehring suggest that trusted platforms do not insist on provable security for all conditions, furthermore the end user may not know, understand and therefore trust any proof presented to them. It is therefore deemed more important that "... a trusted party vouches for the fact that a system configuration and policy is fit for a particular purpose ..." [6, p180] Although we have an interest in trusted platform technologies, the research described in this thesis is not limited to a study of the type of technologies which could be appropriated to construct secure solutions for the military. The target platform is only one part of the challenge; the more urgent and difficult part to be solved is investigating the way in which secure solutions may be manipulated to provide the levels of flexibility demanded by their human operators. Furthermore, it is necessary to incorporate this part of the puzzle at the start of the system design phase so that all stakeholders can make informed decisions about the engineering process armed with knowledge about the risks that will undoubtedly arise. In short, the research interest of the author lies in embracing the principles of fit for purpose security and seeking novel ways in which it can be

incorporated into the system development cycle in order to derive benefit for users with a need to balance flexibility and high assurance.

In our work we use the notion of fit for purpose security to mean that the defined security requirement is met or achieved at the time it is required. From this we can infer that different security controls, perhaps of varying strengths, can be utilised depending on the level of risk, trust, the event or time the control is instantiated. Timeliness is critical since the requirement must be met when required; not before, nor after. This presupposes that all possible eventualities are known and can be accommodated in the planning process which is, of course, quite unrealistic. When designing fit for purpose security it is necessary to recognise that different degrees of flexibility both exist and can be achieved. This can be visualised on a scale which ranges from adaptation to known, changing conditions right through to addressing unexpected and evolutionary events. Given that the defence community operates in a security critical environment we contend that any work on flexible security must be positioned towards the more deterministic end of the scale but with a level of capability to at least handle unplanned events in a coherent and trustworthy manner. This level of flexibility will provide significant advantages over current measures since, in harmony with Ashby's view, it will be possible to deploy a system that can change and potentially will operate for longer before external intervention is required.

Fit for purpose security is underpinned by the synthesis of policy, architecture and mechanism. Policy forms the core component since it defines and is instrumental in the enforcement of rules controlling the behaviour of objects on the system and ultimately reflects the intent of the stakeholders. However, as revealed in later chapters, the security policy is only one aspect of the fit for purpose solution. If the goal of engineering flexibility and a level of robustness whilst minimising the amount of human intervention is to be achieved, it is necessary for changes to security policy to be driven by external factors which are defined as events of interest. These events may include a detected change in the level of trust (when a user joins a different community), increased levels of risk caused by network attacks, loss of a community member and so on. A level of 'intelligence' is required in the system where events of interest can be identified in the environment, processed and actioned thus achieving necessary modifications to the separation and sharing profile of the system. Our fit

for purpose solution must therefore exhibit at least a minimal level of intelligence in order to deliver this capability.

A pragmatic approach has been taken in that the research seeks to use and augment existing technologies and frameworks where possible. Our target community, has a strong motivation to incorporate Commercial Off The Shelf (COTS) based solutions where practical and appropriate in order to reduce cost, avoid 'lock-in' to proprietary solutions and broaden the base of available products from which to select. By recognising this motivation, we have the opportunity to assess the strengths and weaknesses of current solutions and provide enhancements where limitations are identified. Under this regime, technology may be improved for all.

Another aspect of fit for purpose security is concerned with assurance. A level of proof and reasoning about the changes brought about by a flexible security solution is required in order to instil confidence that flexibility does not result in less security or increased and unacceptable risk. This can be achieved through validation.

In [8] McMurran *et al* observed that as systems become more complex and integrated it becomes increasingly difficult to show that the desired properties actually hold. In particular, it is hard to detect emergent properties of a System of Systems (SoS) because the properties of individual systems are not necessarily compositional. The authors continue by describing the application of formal methods to the description and development of large complex systems so that knowledge of individual systems can be used to identify weaknesses in the integrated whole. Although the project is specific to the development of dependable systems, there are similarities with our research on developing fit for purpose security. Part of our approach is developing a security policy that adjusts in accordance with monitored changes in the environment. Since the overall system embraces a number of relationships between different elements of the whole system it is absolutely fundamental to identify the ramifications of introducing flexibility into one or many components. In our work, we use both informal and formal approaches to describe and validate the security requirements of an interconnected system. Both approaches allow us to analyse different states and validate any action taken to mitigate identified risk. Of course it is recognised that a formal method, where the analysis can be automated and repeated by an independent

assessor, will provide greater proof and corresponding assurance that the security policy controls the system in the desired and documented manner.

1.1 Definition of Terms

The research areas covered are broad and complex. The purpose of this section is to describe how the three key terms of *flexibility*, *high assurance* and *fit for purpose security* have been interpreted and applied in this study.

The term fit for purpose is used in many different contexts. Essentially it is a measure of suitability; that is the object under consideration has an intended purpose for an intended consumer. If used as directed for the advertised purpose by the intended consumer, the creator of that object may make certain claims about the expected behaviour, performance or tolerance of the object. Implicit in this statement is a level of confidence and expectation, that is, the intended consumer may reasonably expect and 'trust' the object to behave in a particular manner under certain conditions. There are some interesting tensions in the notion of fit for purpose. For example, in our study of security in agile mission groups, the creator of the system may make decisions concerning what is fit for purpose. However, because they are not the actual user group, the experience of the recipient may be very different. The security applied may not be appropriate for the conditions under which they are operating and furthermore those conditions may change. So, we may ask 'fit for whose purpose?' Central to our work is the assertion that within a system there are different levels of 'fit' dependent on the circumstances at the time security is required. One size does not necessarily fit all. We adopt the view that a system can be deployed with a baseline set of security policies and controls that will be suited to our expected mode of operation. In order to improve the likelihood of security *remaining* fit for purpose, predictions about the future use of that system are made. This allows us to reason about necessary changes to the security requirements beforehand and plan for the appropriate security controls to be in place that will support those changes. In all cases it is necessary to find a balance between those who are designing the security of the system and those who actually have to work with it. This can be assisted through the application of judicious modelling where the requirements of the system are articulated in a medium that can be understood by all stakeholders. If it is assumed

that changes to these requirements will occur, and possibly very locally and rapidly in the case of agile mission groups, it is proposed that this approach provides an improved opportunity for engineering appropriate or fit for purpose security which can be exploited as a business enabler for the target community.

In the security community the term 'assurance' has a very specific meaning however, it is often used quite loosely and out of context which may result in confusion. In [9] assurance is defined as "...the trust that can be placed in a system, and the trusted ways the system can be proven to have been developed, tested, documented, maintained and delivered to a customer...". Of particular importance is the word 'proven'. In order to achieve a recognised level of assurance it is necessary for developers and manufacturers to submit their products to rigorous and independent testing against an established and formally accepted set of criteria. It is this testing process that essentially provides the 'proof' that a product achieves the criteria. Internationally these security criteria differ; for example, the United Kingdom works to the UK IT Security Evaluation and Certification Scheme (ITSEC) [10]. However, in recognition of the importance of international standardisation, the Common Criteria were developed. These are "...an alignment and development of a number of source criteria: the existing European, US and Canadian criteria..." [11]. Publications such as [12] describe the role and process of Common Criteria certification in the UK which is operated by the Communications-Electronic Security Group (CESG) [13]. In order for security evaluation certificates issued by individual nations to be recognised internationally, the Common Criteria Recognition Arrangement (CCRA) [14] was set-up between participating nations.

In an increasingly security conscious environment it is advantageous for a security product to undergo an independent security evaluation programme in order to achieve an evaluation assurance level for the product. In the UK, ITSEC certification levels range from E1 (lowest) to E6 (highest). The Evaluation Assurance Levels (EAL) for the Common Criteria range from 1 (lowest) to 7 (highest). There is correspondence between the levels from E1 and EAL2 onwards, that is, there is no equivalent ITSEC level for EAL1. In [9, p.106] the authors observe that at the higher levels of security there are few changes in the security features themselves but a definite increase in the degree of assurance that a user can place in the system's architecture and security

policies. This is derived from the increase level of rigour and possibly the requirement for a formal approach to be taken in the design and development of the solution. As a result, the user can have greater confidence in the whole production of the product they are purchasing over and above the functionality it provides.

The military is a security critical environment and requires a high level of confidence in security properties such as confidentiality. It is therefore expected that an assurance rating will probably be mandated for any technologies utilised in the system. Yet, this returns us to the question of *balance* between fit for purpose and high assurance since there may be occasions where the implementation and configuration of a high assurance solution actually prevents the user from conducting a perfectly legitimate task. For example, at the start of an operation users may be denied access to particular sources of information because under normal circumstances they would not have a 'need to know'. As the operation unfolds events in the environment may move on; perhaps there is a change in hostilities or collaborative partners. As a result, access to these information sources may now be required in order for the user to deal with new challenges. The initial security parameters are no longer valid because the original assumptions are out of date and the security solution is no longer fit for purpose. Interestingly, a system or product is certified against a specific set of operational parameters. Therefore if the user configures or uses it differently, the level of assurance claimed originally may be invalid.

This research assumes that "assurance" is synonymous with "confidence" without the connotation of an evaluation level. That said, a significant part of our early research focused on separation technologies and architectures which could be employed to engineer a higher level of assurance should it be necessary to adopt the formal evaluation route. Furthermore, the architecture of our solution has been devised with the principles of high assurance and formal evaluation in mind. Ultimately, our interest lies in manipulating those technologies through security policy to provide the level of flexibility in a highly segregated and controlled world. We believe that stakeholder confidence in the solution can be increased if the impact of flexibility and behaviour of the system can be quantified through modelling; particularly if the modelling framework used is underpinned by a formal language. Although in the lifetime of this research our work will not undergo formal security evaluation testing

such as that described in [10], we judge our approach to be a first step in the development of a system that balances flexibility in cases where the user also needs a high level of assurance or confidence in the properties of that system.

It is possible to engineer different capabilities that will introduce flexibility into a system depending on available technology, stakeholder requirements and their attitude towards risk. These levels range from the highly flexible and non-deterministic end of the scale where a system would adapt, perhaps in unexpected ways, through continuous learning and observation of their environment through to a deterministic system that allows adjustment in accordance with a pre-defined policy. In the latter, a system planner would attempt to predict how changes might reveal themselves and design the system components so they are able to change accordingly (perhaps more of a traditional control system approach). According to the Penguin English Dictionary [15] intelligent:

"...is said of electronic equipment: able to respond to change and to initiate action based on that response..."

Our research is intended to have an application in environments where high assurance and minimal intervention with deployed devices are both requirements. It is therefore necessary to position the work at the more deterministic end of the scale, where the system is still pre-planned but allowed to change when the situation dictates. In accordance with the dictionary definition above, we propose that our solution can be described as 'intelligent'.

1.2 Scope

The research was based against the version of DBSy® which was available in 2005. Upgrades to DBSy are an on-going process and current versions at the time of issue of this thesis may be different from the version used at the time of this research. In the opinion of the author there have been no changes that invalidate the interpretation or use of the approach and affect the results published in this thesis.

The experimental work presented in Chapter 6 was based on the Fedora Core Linux distribution, version 6.

1.3 Motivation

Our motivation for conducting this research is two-fold. Firstly we believe that security poses the most significant risk to realising NEC because it is currently unable to accommodate the level of flexibility required between changing and collaborating communities of users. This problem starts at an abstract level with the modelling process because there is no means of including and reasoning about changing requirements at this level. As a result it becomes difficult to implement security solutions that adapt to accommodate operational need. If security hinders quite legitimate tasks because the system is inflexible, then it may become necessary for the user to find a workaround. This activity will not, of course, have been analysed to assess the level of risk that might be introduced as a result.

The second part of our motivation stems from the need to ease management overhead in our target community. The ability to model and measure the impact of changing security requirements allows us to engineer more robust systems. The system can be designed with options and the means necessary to adopt those different options when the right conditions prevail. We must consider that our target audience frequently operates with limited resources and in a hostile environment. It is desirable to minimise the amount of effort they must expend on system planning. Our fit for purpose security solution arises from delivering appropriate security when needed and ideally with minimal intervention from the human user.

1.4 Contributions

The main contribution of this thesis is the development of a methodology which enables us to realise fit for purpose security from high-level modelling through to implementation. The methodology is underpinned by the synthesis of security policy, architecture and mechanism and delivered by two new concepts the Non-Persistent Capability Concept (NPC2) and Non-Persistent Domain (NPD). Section 1.4.1 introduces these concepts.

1.4.1 The Non-Persistent Capability Concept and Non-Persistent Domain

The NPC2 and NPD are both used to capture, articulate and measure temporary changes to the security profile of the system. Our work with the NPC2 has evolved further and includes an implementation framework and practical demonstration of the concept in a prototype solution. Different chapters of the thesis discuss the individual uses of both constructs which, by way of introduction, are defined in the paragraphs below.

Modelling Security Requirements

In the military context security is synonymous with separation. When developing the security policy for a system it is reasonable to begin the modelling process with some expression of the separation and sharing requirements. An approach and notation developed specifically for the task of documenting such requirements in the defence community is the Domain Based Security Architecture DBSy® from QinetiQ [16], [17]. Since our interest is focused on, but not limited to, the defence market an analysis of the DBSy approach is fundamental to the research. It provides an established and comprehensive environment in which to present our work on fit for purpose security models. Closer inspection of DBSy led us to believe that it was appropriate and efficient in environments of a relatively static nature and its primary function was to capture and articulate the maximum permitted connectivity of a system. Given that our interest lies in the development of fit for purpose security to enable changing requirements in a system, we hypothesised, correctly, that DBSy would exhibit limitations when applied to this type of model.

In order to address what we considered to be a limitation and add to the expressiveness and potential usefulness of DBSy, the thesis proposes some simple additions to the DBSy notation which are embodied in the NPC2 and NPD constructs. In the DBSy models, the NPC2 shows where alternative separation and sharing options exist in the system. The notation allows the system architect to distinguish between the baseline, that is, 'normal' condition and exceptions to the baseline. The NPD was developed to capture the existence of a temporary community which could be, for example, an ad-hoc network occupied by some members of the existing

system(s) and/or third-party agencies who are active members of the community for a limited period and only under certain conditions. The NPD could provide the means of limiting information sharing where it is necessary to temporarily include an external community with a lower level of trust. Other members of the NPD may have alternative business connections set up with regular domains which are normally used to share information. Such examples are described in Chapters 3 and 7.

It is our view that models employing either the NPC2 or the NPD or both constructs provide a better representation of the security policy requirements. Moreover, their use alerts the system designer to areas where it will be necessary to architect flexibility into the security solution. If the user is provided with more comprehensive information regarding the changing architecture of the system and how this impacts risk, it is concluded that greater confidence in the integrity of the system can be achieved under all of its states.

Risk Assessment

The security modelling process includes an analysis of risk inherent in the models. DBSy risk assessment is concerned with (but not limited to) analysing the permitted business connections between domains which are necessary to enable sharing, but could also be used to compromise the system [17]. Although DBSy identifies and includes a number of different system threats as part of the risk analysis approach [18], for the purposes of this research it is unnecessary for us to adopt all of them. Our risk assessment assumes that connectivity is also a measure of vulnerability. Given that the NPC2 and NPD constructs are deliberately designed to affect connectivity on a temporary and/or conditional basis, we are particularly concerned with assessing their impact on the risk profile of the system. It is therefore sufficient, at least in the first instance, to focus on the existence of the permitted business connection between domains and the role of the NPC2 and NPD constructs and omit individual threats associated with those connections as defined in the DBSy approach. It should be noted that as part of the risk assessment process the analyst would usually select security controls to mitigate identified risks and then reassess the system to calculate their effect. Since our study focuses on connectivity and does not include individual threats in the scope, it is inappropriate to follow this aspect of the risk assessment process and investigate the type of controls that may be applied.

The NPC2 allows us to identify all of the sharing requirements that it is believed *may* be required during the lifetime of a deployment and conditions on those capabilities which may be exerted. The NPD provides further refinement where a sharing requirement may be between a regular domain and an ad-hoc community. Both constructs impact the separation and sharing profile; however they do so in different ways. The NPC2 is ultimately responsible for permitting or denying the sharing of information under different conditions and it therefore forces a choice to be made. Essentially it can reduce the number of potential paths because it acts as an exclusive OR between domains of interest rather than assuming that both are active. Furthermore, the NPC2 distinguishes between the baseline and exceptional policies making it possible to identify the number of compromise paths we could reasonably expect when the system is operating under normal conditions. Exceptions may never occur. However, in the event that they do, the impact on the separation and sharing profile of the system and subsequent risk can be calculated.

On its own the NPD does not place conditions on connectivity. It simply reflects the case where a temporary community will be set up, although of course it could be used in conjunction with the NPC2 if the security policy limited participation in an ad-hoc network when other sharing capabilities were enabled. By recognising its existence, the impact on potential compromise paths in cases where the NPD is in use or is inactive may be calculated.

To assist with the risk assessment process, the DBSy compromise path models produced for our scenario are redrawn using graph theory notation [19, p.11]. Although this very simple view of connectivity does not allow the capture of flexible (or non-persistent) connectivity it is convenient in providing a coherent transition between graphical representation and mathematical analysis of potential compromise paths using matrices [20, p.4]. As we show in Chapters 4 and 7, the NPC2 in particular displays some interesting characteristics when modelled as a matrix, particularly as we begin to calculate higher order connectivity arising from multiple 'hops' between domains of interest.

Implementation

The NPC2 is the more mature of the constructs and has been implemented and tested within a demonstration system. A conceptual architecture has been developed for the NPC2 which is presented in Chapter 5. This framework can be applied by the systems engineer to construct a working solution. The model describes the individual elements comprising the NPC2. This includes the functionality and services provided at each level and the interface with other layers in the architecture. As such it provides a necessary link between the theoretical discussion communicated in the first half of this thesis and the practical demonstration of the concepts in the second half. To conclude the study and support our claim that the NPC2 can be applied at all levels, a working prototype constructed in accordance with the NPC2 architecture, is documented in Chapter 6.

Security Patterns

Security in the discipline of Information Technology has attracted active research participation for decades. In [21, p.31] a *security pattern* is defined as:

"...a recurring security problem that arises in specific contexts and presents a well-proven generic solution for it. The solution consists of a set of interacting roles that can be arranged into multiple concrete design structures, as well as a process to create one particular such structure..."

In the context of this definition and following a brief review of some existing patterns described in [19] it is proposed that the NPC2 does exhibit properties of a security pattern. The NPC2 architecture described in Chapter 5 is a technology independent and modular framework which has been created to solve the problem of enabling flexible security in a very specific context; high assurance environments. In the limited duration of this research activity the NPC2 has not been subjected to the rigorous application normally demanded of a concept before it is accepted as a pattern. The potential for describing the NPC2 as a security pattern was discovered late on in the research and as a result, there has been insufficient time to pursue the idea through to a logical conclusion. It is deemed to be of sufficient interest to note as part of our contribution and an area that should be actively pursued as future research.

1.4.2 Publications

In addition to the development of the NPC2 and NPD as described in this thesis, our work has also contributed to the literature in this area. Some of the material in this thesis is included in the publications listed.

H. Janicke and L. Finch, "The Role of Dynamic Security Policy in Military Systems", *Proceedings of European Conference on Information Warfare*, Shrivenham, UK. July, 2007, pp.121-130. [22]

H. Janicke and L. Finch, "The Role of Dynamic Security Policy in Military Systems", *Journal of Information Warfare*, Vol. 6.3, UK. December, 2007, pp.1-14. [23]

L. Finch and R. Vaughan, "Towards Fit for Purpose Security in Military Systems", *Proceedings of European Conference on Information Warfare*, Plymouth, UK, June 2008, pp.71-80. [24]

1.5 Thesis Structure

This thesis documents the author's primary theme, fit for purpose security, as a journey of two halves. The first part of the thesis applies the NPC2 and NPD constructs to the security modelling and risk assessment processes. In doing so, attention is focused on the theoretical implications of capturing and modelling fit for purpose security. The second part of the journey documents the practical work conducted to implement a working implementation of the NPC2. This covers the development of the NPC2 software architecture and concludes with a description of the practical example which was constructed to demonstrate one application of the NPC2.

The work documented in this thesis has been conducted without the membership and associated support of a research group that would normally be expected in post-graduate studies. The author has therefore played a critical and central role in opening and developing the research argument. Throughout the thesis the personal pronoun 'we' or possessive pronoun 'our' is used. However, it should be noted that the concept of the NPC2 and NPD and its application through the modelling, risk assessment and

engineering processes is entirely that of the author. During the course of the research, the author formed relationships with other individuals to explore the concept of the NPC2 in a wider context. One example of this collaborative work is described in Chapter 3 where the author carried out some investigative work with DeMontfort University regarding the formalisation of output from the DBSy models. The author's supervisor has also been instrumental in supporting further work with the connectivity matrices in Chapters 4 and 7. In addition to the work expressed in this thesis two joint-author conference papers, which develop the ideas, are currently in production. As described in Chapter 6 a small team of two people, the author and a software developer carried out the actual development of the NPC2 example. At this stage the author became project manager and solution architect. This involved producing the NPC2 architecture, designing the solution that would demonstrate the NPC2 and fulfil the requirements of the case study and solving the problems which naturally arose from developing an example in a resource constrained and demanding environment. Under the guidance of the author, the software developer produced or adapted the necessary software code to realise NPC2 functionality.

The remaining sections of the thesis are structured as follows. Security related research has been ongoing for over three decades therefore available literature related to the subject is plentiful. A selection of related work is presented in Chapter 2 which concludes with a statement of the problem to be addressed by this research. Chapter 3 embarks on the primary theme of the work and begins with the informal modelling process using the DBSy graphical approach and notation. A small-scale illustrative example has been devised which allows us to apply DBSy and assess our proposed additions to the notation. The research is not limited to an informal modelling approach and in the last part of Chapter 3 output from the DBSy models is utilised in a formal modelling approach developed at De Montfort University. This study was conducted as a collaborative exercise by the author and Dr. Helge Janicke of De Montfort University and resulted in two joint publications listed in Section 1.2.2.

Chapter 4 looks at risk analysis and applies aspects from DBSy to our scenario. Our objective is to assess the impact on system vulnerability resulting from the introduction of flexibility via the NPC2 and NPD constructs. As part of this exercise we begin examining existing theory concerned with the rate of growth and

proliferation of connections as domains are added to the network. This discussion is also included in Chapter 4.

Chapter 5 links the theoretical aspects of the work to the practical implementation phase and describes a high-level architecture for the NPC2. Chapter 6 contains the experimental results from applying this architecture to a working prototype of the NPC2. This work covers one way in which the NPC2 may be engineered; there are of course many others. The examples presented in this thesis are small-scale by design. This is necessary in order to keep the research manageable and within scope. Alternative applications for the NPC2 and NPD constructs are described in Chapter 7. This provides evidence that the concepts can be scaled both in terms of modelling and in the case of the NPC2, this includes implementation. Finally Chapter 8 reflects on the success of the research and presents our conclusions. Like most research endeavours it has not been possible to address all of the issues which have been revealed throughout the programme. We therefore present the areas which are outstanding but in our view, constitute valuable opportunities to continue evolving this important area of work and assist with further developing fit for purpose security for flexible but high assurance environments. This discussion concludes Chapter 8 and the thesis.

Chapter 2: Literature Review

The development of information systems in environments, where high levels of confidence are a prerequisite, is dominated by the requirement to construct provably secure systems. Research which investigates *how* this can be achieved has interested the academic and commercial communities for almost four decades and continues to stimulate discussion. This well established area of research has generated a correspondingly large amount of literature therefore it is necessary to be selective in our choice of work in this review. A number of reports are available which eloquently summarise security models in general, including [26], and policy specification in particular [27]. It is not our intention to repeat these works; instead we extract and critique aspects of particular interest to our research.

Our interest lies in enabling both flexibility and security in high assurance communities, and investigates *how* this may be engineered into a system through the synthesis of security policy, control mechanism and architecture. We also employ the principles of a separation architecture in that we define both the elements of the system that should (normally) be separated from one another and the ways in which sharing should (generally) occur. The emphasis is on the baseline or normal security requirements since through our research the system evolves to accommodate exceptions to the general pattern of behaviour. This provides flexibility which is necessary to support changes to the separation and sharing profile.

During the course of our literature survey it has become apparent that an important contradiction exists between flexibility and high assurance in computer systems. The former suggests the capacity to adapt which will result in emergent and potentially unexpected behaviours with the capacity to undermine the security of the system. High assurance of course is normally associated with tightly constrained and deterministic systems. However, as observed by Kuhlmann and Gehring, [6, p.180] these designs tend to counteract any advantages to be gained through flexibility. This dichotomy is important for our research since it is necessary, in our view, to develop systems that can support the changing information and resource requirements of communities such as the military. At the same time a level of assurance must be

preserved to demonstrate that flexibility is not achieved to the unacceptable detriment of security. As stated in Chapter 1, our research is inspired by the work of Kuhlmann and Gehring and whilst they focussed on the role of trusted computing platforms in digital rights management, there is an important parallel with our research. The authors proposed that this type of technology could offer a fit for purpose solution that would seem to provide a level of balance between flexibility (and openness) versus the need for a closed system. Furthermore the authors recognised that the balance between openness and flexibility would be dependent on the needs of the system owner and their communicating peers. However, from our perspective the more important observation was that this balance would be at a *given point in time.* We extend this idea to assert that security requirements of the user will change in response to time, event or level of trust: therefore fit for purpose security must take into account these stimuli.

Early research into security models and frameworks was significantly influenced by the military primarily because significant funding was invested by this community during the 1970s [28, p.137]. The military have a clear need to preserve, perhaps above all else, the confidentiality of information. However, the multi-level concepts developed to support this requirement are also applicable to enabling data integrity; a property of particular interest in commercial environments [29]. Although there are some differences in the classifications across nations, Governments apply a fairly standard protective marking scheme to information graded from Unclassified through to Top Secret and higher. These classifications provide the criteria on which to separate (and control access to) resources. When considering confidentiality, a subject may only read an object if his or her security clearance is at least equal to the protective marking of that object. This structure of security labelling and access based upon clearance defines multi-level security models. The challenge for computer scientists, particularly in the 1970s, was to define and enforce multi-level security into computer based systems. A study by Anderson [30] proposed the concept of a *reference monitor*, a component in the operating system designed to mediate access to resources but which was also sufficiently small and simple to undergo rigorous scrutiny and testing and provide some level of confidence in the trustworthiness of the component. Functionality in computing devices is constantly increasing with a corresponding growth in the complexity and size of the operating system. In short, it

has become increasingly difficult to verify mainstream operating systems. During the 1980s Rushby and Randall [31] observed that attempts to construct secure general-purpose operating systems had not been notably successful. They maintained this was due to the reliance on a security kernel as the primary mechanism for enforcing security and proposed an alternative approach which focussed on a distributed secure system rather than a secure operating system. Simply put, the responsibility for security was given to a number of small (trustworthy) components that were interconnected with larger "untrusted" host systems. One of the prime tasks for the trustworthy components was mediating communication between the untrusted hosts. The authors claimed this approach was "demonstrably more secure" and efficient.

In recent times the Multiple Independent Layers of Security (MILS) architecture has gained attention, particularly in the development of secure single purpose applications for security and safety critical environments such as the Avionics Industry. MILS, as its name suggests, is based upon a layered architecture and according to Alves-Foss *et al*, [32] it is possible, through separation, to construct a hierarchy of security services. Each level is responsible for its own security domain and uses the security services of the layer below to add new security functionality which can be exposed to the higher levels. The heart of the approach lies with a small separation kernel like the one proposed by Rushby in [33]. Functionality that would normally be found inside a traditional operating system is moved up into middleware layers away from the kernel so that it only has responsibility for maintaining separation between resources (in accordance with a pre-defined policy). This MILS approach has been used successfully in current Real Time Operating Systems (RTOS) products from LynuxWorks [34] and Green Hills Software [35]. Under this regime it is possible to achieve higher levels of assurance for the RTOS which is necessary for military applications or other environments with security or safety critical requirements. Products based on MILS do present certain limitations for our research however; the primary one being a static security policy. The target system utilising a separation kernel would not be able to adapt to changing circumstances without restarting the system with a different policy, which rather defeats the objective. Nonetheless, we consider the approach a very interesting development and propose that the role of dynamic policy within this context is worthy of future research.

During the 1970s researchers were actively investigating how multi-level secure systems could be described and modelled for computing systems based on mathematical proof. Probably the most influential security policy model to emerge at this time was the Bell and LaPadula (BLP) confidentiality model [36]. The model is designed for the military environment and thus relies on an appropriate labelling scheme which reflects military interests. Labels describe the protective marking of an object, for example Secret, Restricted, Not Protectively Marked, and the security clearance of the subject who wishes to access it. In the BLP model two security principles are enforced which are defined as 'no read up' and 'no write down'. In short the subject is not permitted to read information labelled at a higher protective marking than their own clearance nor are they able to write an object with a higher protective marking to a lower protective marking. Although Bell and LaPadula were able to prove the model mathematically it did exhibit certain limitations which were identified by other researchers. Biba [37] for example, proposed an alternative model to accommodate integrity rather than confidentiality. Clarke and Wilson [27] were interested in preserving integrity but rather than focus on the defence or military arena like Biba, they developed their model for the commercial environment. Brewer and Nash [38] were also concerned with confidentiality but unlike Bell and La Padula their model focuses on the commercial context.

An additional concern lies with the practical realisation of the aforementioned multi-level security models. In a practical environment, the BLP model, without refinement, proved operationally difficult to implement and use. Furthermore, whilst static security policy has appeal because of the ability to prove security at given states it also gives us cause for concern. NEC requires interoperation and collaboration between changing communities of users who have different security characteristics, levels of trust and information needs. Whilst multi-level security models can successfully capture the various classifications of resources in such a situation, on the face of it, there is no provision to accommodate changes in the security profile that may be necessary once the system is in use. Bell and LaPadula introduced a refinement to their model in an attempt to address the usability problem resulting from such a strict definition of security rules and accommodate, to a limited extent, the concept of change. This refinement is the *principle of tranquillity* which, as described in [39, p.143] takes two forms. *Strong tranquillity* [39, p.143] specifies that

a security label never changes during system operation. *Weak tranquillity* [39, p.143] is more interesting and relevant for our work with fit for purpose security. This form states that security labels do not change in such a way as to violate the defined security policy. In other words, the restriction is modified to allow what Bishop refers to as 'harmless' changes to security levels [39, p.143]. In practice, a trusted entity or trusted subject is permitted to sanitise and declassify the object to a lower security level. In this way, the Bell and LaPadula rule of 'no write down' is upheld since through the process of sanitisation sensitive information, that would normally warrant the high security classification, is removed before writing down to a lower classification.

Although weak tranquillity is a more practical and realistic solution in real-world systems it does not fulfil our vision of fit for purpose security. For example, in the military environment it is necessary to consider operational context. Under certain circumstances it is quite probable that both resources and the opportunity to manage information will be quite limited, therefore applying the principle of weak tranquillity in this situation may simply not be practical. The violation of the 'low' classification principle where a subject has temporary capability to access 'high' classification resources may, in specific short term instances, be an acceptable fit for purpose alternative. Our research explores this idea in greater detail.

Throughout this thesis the view is adopted where under 'normal' circumstances a subject has restricted access or restricted capabilities in a system. However, under exceptional circumstances, the security policy allows the user elevated privileges for either a specific time period or until events in the system prompt another change in security policy. This approach differs from the weak tranquillity principle in that the classification of an object or subject is not modified. Instead the security policy is designed in such a way to allow temporary changes in the capability of the subject that would normally be associated with a more privileged user. This approach potentially limits the impact of the change quite significantly. As Bishop observes in [39, p.142], when an object is lowered in classification, 'low' subjects (of which there could be many) suddenly have access to high level information (unless of course it is sanitised prior to declassification). In our approach, there is potentially greater control over *who* has elevated access since the change affects the *subject*. Moreover,

escalation of privilege is only instantiated under pre-defined and specific conditions. The system would normally be operating under restricted conditions. It is feasible that conditions requiring a change in capability may not present themselves. However, if they do, then our approach allows the system to accommodate change and thus survive for longer without intervention from an administrator. This approach complements the Law of Requisite Variety proposed by Ashby [5] and enables us to achieve a design objective in system robustness in addition to flexible security rules.

In order for security to be fit for purpose it clearly needs to meet the needs of the system owner and operators and satisfy the requirements of technology experts. A common language would help to avoid misunderstandings between these stakeholder communities. During the 1990s UK MoD sponsored research into developing an approach that could be used to capture and express security requirements in a manner comprehensible to both business and security specialists. The Domain Based Security Architecture (DBSy) approach from QinetiQ [16] originates from this work. As part of this approach a series of models are produced that reflect the sharing and separation requirements necessary to support the needs of the business and security. These models are the Information Security (InfoSec) business, infrastructure and architecture models respectively. The document set produced as part of the DBSy process is intended for use by the appropriate authorities to support the system evaluation process. Since our research includes the practical application of DBSy to a case study, a detailed description of the approach and models are provided in later sections of this thesis.

In [40] Hayat *et al* concluded that whilst DBSy did not explicitly support the modelling of mobile and ad-hoc communications, the benefits of doing so (which would require extensions to the DBSy technique) were of limited value. However, as a result of our research we tend to disagree with this view. Increasingly the relationships between users of either one system or a system of systems are becoming more flexible and prone to change. Moreover, this is the more 'static' end of the mobility scale. The other extreme is the impromptu, ad-hoc community which may or may not be required during the lifetime of the communications but must be included as a discrete part of the model nonetheless. If we cannot capture requirements for flexible security and ad-hoc communities in the DBSy models and risk analysis

process, the ability to provide appropriate security is impeded. It is suggested therefore, that appropriate notation and recognition of these special types of systems is a fundamental requirement. This thesis describes our proposed notation, the NPC2 and NPD constructs which are both described in detail in Chapter 3.

Risk assessment or analysis is a large topic in its own right and the level of detail in its practice will vary across different types of organisation depending on their legal requirements, the value of the assets and the hostility of the environment. Clearly risk is a major concern for the defence industry since a breach of security could result in loss of life. For the purposes of this research it is particularly important to study the impact of flexibility on the security profile of the system. Fit for purpose security is the balance of flexibility with a level of assurance that the resulting risk is not increased or unacceptable. According to [41, p.8] in the field of Information Security risk can be defined as "... a function of the level of threat, vulnerability and value of the information asset ..." As part of the risk assessment process a metric would be attributed to each of these properties and used to calculate the overall weighting for the risk. The process can become very complex particularly in large and richly-interconnected systems. Software tools such as the Central Computer and Telecommunications Agency (CCTA) Risk Analysis and Management Method (CRAMM) cited in [41, p.233] have been developed to assist with risk analysis. CRAMM is still owned by the UK Government and is the "government preferred" method of risk analysis. According to Jones and Ashenden [41, p.234] Insight Consulting [42] is the sole licensee for CRAMM and the tool is used widely around the world. CRAMM provides a structured and rigorous methodology for risk assessment. As a result, the software is itself complex and requires a certain level of expertise which may dissuade individuals who only perform risk analysis on an occasional basis from using it.

The DBSy risk analysis approach analyses the InfoSec Architecture models to look at potential compromise paths arising from the permitted business connections between domains [18], [19]. The method also includes an assessment of threats and vulnerabilities arising from, for example, malicious or accidental compromises caused in the environment or the infrastructure as well as the method used for information sharing. Following a detailed analysis, which would also include an assessment of the

likelihood of compromise (based on the motivation and capability of the attacker), it is possible to determine the level of risk. For the purposes of our research it is unnecessary, at least to begin with, to perform a detailed analysis which considers the motivation of attackers as well as other factors. Of primary concern is the effect of the NPC2 or NPD on the separation and sharing profile which is represented at the most fundamental level through the permitted business connections. How does/do the NPC2 and/or NPD contribute to the level of risk in this context? Do they reduce it? To answer these questions it is necessary only to consider risk at the level of connectivity. If the NPC2 and NPD are successful in reducing the number of compromise paths then our hypothesis is that the affect will be increased when other factors are included in risk calculations.

The DBSy approach has no means of differentiating between different *states* of separation and sharing in a single model. Therefore all potential connections are included in the risk calculations whether they are active or not. The risk strategy is therefore based upon maximum permitted connectivity or the 'worst case scenario'. In the opinion of the author, this is not a true reflection of the risk profile. A more faithful representation of *actual* risk occurs when the non-persistent connections are included in the model. Although it is necessary to perform risk calculations for each of the states of the NPC2, as described in Chapter 4, the task can be automated using appropriate (and potentially trustworthy) software. Not only does this approach reduce the amount of effort and potential for error, it also supports repeatability of the calculations by an independent assessor. The results obtained when the NPC2 and/or the NPD are included in the risk analysis are more complete. It is possible to identify the level of risk introduced under certain conditions and take a more informed decision regarding the type and strength of security controls necessary when the conditions are invoked.

In [41, p.4] Jones and Ashenden observe that:

"...as IT infrastructures become increasingly intertwined, we need ways of carrying out IT security risk assessments that are lightweight and flexible yet allow us to understand interdependencies of risk in a complex and dynamic environment..."

Our work on risk assessment in Chapter 4 seeks to address these issues. Firstly, the effect of the NPC2 is included in calculations for both direct and higher order connectivity arising from indirect (multi-hop) communication between domains. Our models include external domains to represent other systems which may play a role in the case study and impact the security of the agile mission group under scrutiny. Interestingly, connectivity with these external domains may be subject to conditions which are captured with the NPC2. Secondly, it was mentioned in the previous paragraph that software could be used to automate the computation of potential compromise paths. Connectivity matrices are used in both Chapters 4 and 7 to capture the state of a connection. Using matrix multiplication it is possible to calculate higher order connectivity and investigate the relationships between domains. This functionality can be programmed using established tools or languages and for the purposes of this research MATLAB [43] was selected. Not only is it possible to model larger and increasingly complex matrices using MATLAB the calculations can be repeated and validated by an independent assessor which would assist any formal evaluation of our approach or solution. The matrices can include Boolean variables and other values which enable us to calculate and reason about the affect of the NPC2 alongside probability and risk weightings *within a single matrix.* Matrices are commonly used in the assessment process to compare properties such as likelihood of attack, against another property such as impact. That is, they are not normally numeric but textual and descriptive. The Australia/New Zealand model [44] is a prime example of this approach. Connectivity matrices [21, p.4] are commonly used to express the mathematics and language of graph theory [45]. Graphs and matrices are used in our analysis to show the richness of connectivity and the accessibility of domains. Quite simply, large values in the matrix indicate a highly connected or accessible domain. In our view this can also be indicative of the potential vulnerability between different parts of an interconnected system because the same permitted connections that are used for legitimate business operations could be used as a means of launching an attack on the system. It is concluded that the approach we have adopted which uses MATLAB to calculate connectivity matrices is totally in harmony with Jones' and Ashenden's aspiration for a lightweight and flexible tool that can be used to assist the risk analysis process.

The discussion concerning maximum permitted connectivity and actual connectivity continues as part of our investigation into the rate of growth in potential compromise paths when domains are added to the configuration. Of course the increase will be dependent on the number of permitted business connections between new and existing domains. We also contend that the temporal and conditional aspect of connectivity should be included in the growth calculations. It will be shown in Chapters 4 and 7 that our conclusions concerning the rate of growth differ from the work of Metcalfe [46], Briscoe, Odylzko and Tilly [47]. Unlike Metcalfe we do not regard connectivity as a measure of *value* but a measure of *vulnerability* since a connection could be used as a means of compromising the system. A domain with many connections may therefore be more vulnerable than a sparsely connected domain. If the rate of growth when domains are added to the network is as generous as all of these authors suggest, then we judge the situation does not bode well for risk. However, through the use of the NPC2 and indeed the NPD construct we believe that a less pessimistic view of potential compromise paths can be achieved and the rate of growth of connectivity and risk can be substantially reduced. As seen in Chapters 4 and 7, the use of the NPC2 and NPD yield interesting and in some cases counterintuitive results regarding the rate of growth when the matrices are multiplied to calculate higher order connectivity. The reduction in potential compromise paths becomes increasingly obvious as the number of hops between domains is increased. This lends support to the usefulness of the NPC2 within a system, not just to represent changes in the separation and sharing profile that are required for normal operation, but also as a tool that can be used quite deliberately in the configuration to reduce the connectivity (and vulnerability) of a domain at a given point in time. This approach underpins our philosophy of fit for purpose security.

DBSy offers significant advantages to the user community because of its intuitive, graphical approach. It was not developed as a formal approach and provides only a high level view of the sharing requirements to support the business and separation requirements to support security. Our issue with DBSy lies in its *static* view of these connections where in reality, they will undoubtedly change. In related work [23, pp. 125-128] we suggest how output from the DBSy models can be further refined using a formal security policy framework. For this purpose the Security Analysis Toolkit for Agents (SANTA) [48], [49] and [50] was applied to the case study described in

Chapter 3. Not only does SANTA provide benefits in that a greater clarity of expression can be achieved through the linguistics of the tools, given that it has a formal underpinning in Interval Temporal Logic [51] it also affords a level of formal proof that cannot be achieved from the graphical models. Crucially for our work in fit for purpose security, SANTA allows us to capture and express dynamic security requirements. In [23] we provide examples of how the DBSy approach and SANTA framework can be combined and show how the policy can be validated using the Security Policy Analysis Tool (SPAT) [52].

Our vision of fit for purpose security encompasses security policy, mechanism and architecture. Therefore the study of policy frameworks, capable of capturing dynamic properties, is only one aspect of the work. In order to realise a demonstrable example it is necessary to look at how properties from the security policy models have been realised into useable and workable components that can be employed to build secure, but flexible systems.

The foundation of a system is the operating system. This is fundamental to the security of a system since it has responsibility for mediating between applications and the underlying system resources and enforcing the security policy of the system. Without such a secure foundation any attempt to create security through applications alone will be like a 'fortress built upon sand' [53]. Loscocco *et al* [54] continued the discussion on secure operating systems. They concluded that there was an assumption in software development that applications could be created to enforce security without the support of the underlying operating system. This approach, the authors noted, could result in system-wide vulnerabilities. A good mainstream solution was required and this is where the security enhanced version of Linux (SELinux) makes a promising and positive contribution. We do not intend to provide a detailed account of SELinux or its origins in this literature review. Instead we would recommend that interested readers consult the National Security Agency [55] website where there are a number of papers describing the evolution of SELinux. In this review we provide a very brief introduction to the technology which is necessary to appreciate how we intend to utilise and extend existing work to illustrate our own research goals in fit for purpose security.

SELinux implements a flexible mandatory access control mechanism called Type Enforcement [56]. Under this regime, each subject and object on the system is assigned a type identifier. In order to gain access to an object, the subject's type must be authorised for the object's type irrespective of the user identity of the subject. SELinux provides a high level of granularity since a type is assigned to individual objects on the system. Furthermore the system administrator has complete control over the labelling of resources and development of security policy rules to suit their own organisational and security requirements. Although this level of flexibility is potentially more appealing to the wider community there are of course some disadvantages. In this case, one of the downsides for SELinux, particularly in the early days, was and still is the difficulty in developing security policies. However, as Mayer *et al* point out in [57, p.240]:

"...SELinux policy is 'rich' and complex; necessarily so because SELinux provides fine-grained access control for the rich and complex Linux kernel and its interactions with the multitude of user space applications..."

In other words, SELinux does not, in itself, add complexity. As we have found during the course of our research however, this knowledge is of little comfort to the novice policy developer. What is more helpful is the vast amount of interest and activity from the SELinux community in developing strict [57, p.242] and targeted [57, p.261] example policies that will suit the security requirements of many standard systems and which are available in the mainstream SELinux distributions. Furthermore there is ongoing work in developing a more modular approach to policies in the Reference Policy project [58] and policy development frameworks to assist with writing policy [59]. In our work, we are interested in how this seemingly flexible approach, incorporating the strength of mandatory access control with the highly configurable benefits of type enforcement, can be exploited to support fit for purpose security where it is necessary to enforce conditional policy based on (pre-defined) conditions.

The SELinux policy language was extended in 2004 to include conditional policy language extensions [60]. This provides the capacity to define conditional rules where the definition of the rule is dependent on the value of a pre-defined Boolean variable. For example, a subject may only be permitted to run a particular process if

the associated Boolean variable is set to True. Under normal circumstances, and let us assume that the default setting for the Boolean is false, the subject would not have the required privileges. However, if its value is true, the policy contains the necessary rules to allow the process to be run. It is possible for a 'user' occupying a suitable role to change the value of Booleans in a running system. Therefore the policy can be written such that behaviour changes 'on the fly' that is, without restarting the system. This has the obvious advantage that disruption to the system operator is minimised, which in a system critical environment, such as the military, would probably be essential. Of course such an environment would also demand that whatever is used to change the value of the Boolean can be trusted. This is even more critical where we want to minimise human intervention and thus have a background process changing the value when a condition is met.

The use of SELinux to support adaptive conditions has been explored by others in recent literature. In [61] Gregory and Loscocco developed a prototype system to demonstrate support for Risk Adaptable Access Controls (RAdAC). The overall objective of this work is similar to our own, since they are seeking to secure a change in the security profile of the system using SELinux policy. However, the most striking difference between our work and theirs is that our policy change is carried out automatically through the use of a Boolean setting and conditional policy statements. Gregory and Loscocco's work uses a graphical user interface (GUI) from which a system administrator is able to select a new policy to load, based on their perception of risk. The authors observe that real applications supporting RAdAC may require a more automated means of identifying the need to change security policy. They suggest this could be done by replacing the manual "risk knob" with external sensors in the environment to detect when a new policy should be loaded. Our work provides a demonstration that this extension is possible. The security policy automatically makes changes to a user's capability based upon the detection of a pre-defined condition in the environment. The NPC2 architecture defines an environmental monitor component which could be a simple software process (as used in the demonstration system described in Chapters 5 and 6) or a combination of external sensors (hardware) and software. Such implementation specific details are completely dependent on the requirements of the target system since the NPC2 does not dictate the type of technology used.

The Dynamic Policy Enforcement Agent (DPEA) described by Pollet, Butler and Hale [62] is probably the closest rival to our work to emerge in recent literature. According to the authors, the DPEA tool provides a new mechanism using the SELinux security policy to deflect attacks. To achieve this goal, the tool monitors the environment for internal and external threats and uses SELinux Booleans coupled with conditional policy statements to modify the security level of the system. Our work is similar in that we incorporate the same constituent components: environmental monitoring, some form of policy trigger mechanism (which can be realised through Boolean variables) and conditional statements. In our research these components when combined form a demonstrable implementation of our NPC2 which forms the basis of this thesis. However, the NPC2 differs from DPEA in that we are interested in *enabling capability* at the time it is required and in response to specific events. Interestingly the NPC2 could also be used to defend the system (by enabling a different or potentially more limited set of capabilities) but its ultimate objective is in supporting fit for purpose security to the system owner or user. We utilise the security design goal of SELinux, that is, 'deny all' by default and through standard security policy a baseline set of capabilities are permitted that will enable the user to operate under 'normal' conditions. The NPC2 is designed to cope with exceptions to the baseline and allow the system to 'adapt' to changing conditions when those conditions prevail. This is achieved through the conditional policy statements. Furthermore, our NPC2 is a concept that can be applied at many levels. In this thesis we show how it is used at the system modelling stage to represent conditional and temporary behaviour in the target system and as a means to assess potential compromise paths during the risk assessment process. In Chapter 5 the NPC2 architecture is specifically described. Our prototype system using SELinux is only one example of how the NPC2 architecture can be constructed. Since the NPC2 architecture itself makes no assumptions about technology, system developers have control over the components they use to construct each element of the NPC2. Conversely, the DPEA example appears to be bound to a very specific technology platform. Finally our work with the NPC2, whilst offering a flexible solution, is without doubt more biased towards the deterministic end of the flexibility scale. Given our target environment, this is considered appropriate. However the NPC2 architecture is not a limitation since it can be implemented in other ways which would support a less deterministic approach.

2.1 Statement of Research Problem

Following our review of relevant literature in the complex and vast area of information security, it is our belief that research in developing a more flexible and dynamic approach to security through policy, architecture and mechanism is still an emerging area with unsolved issues and the potential for further investigation. Our research is grounded in the context of the following question:

If current security policy models do not adequately express or capture the security requirements of high assurance, ad-hoc and collaborative interactions, what options are available for engineering solutions appropriate to the Network Enabled Capability military initiative?

In addressing this question, our research has followed a systematic appraisal of security policy, architecture and mechanism since it was discovered early on that a solution would require a synthesis of the three. Our initial investigations into this question led us to believe that a feasible answer lay in developing fit for purpose security solutions that could be applied to the NEC environment. The remainder of this thesis describes how we approached and achieved this goal and thus successfully addressed the research question.

Chapter 3: Modelling Changing Security Requirements

The security policy itself is central to developing an appropriate security solution. The development process begins with a high level expression of requirements. At the outset these may be simple and relatively conceptual natural language statements which can be subsequently captured and made more specific within a modelling framework. Through a gradual process of refinement, an increasing level of detail and possibly formalism will be applied to the statements until rules that can be implemented as 'policy' in the target system, are defined. It is at this stage that the policy becomes suitable for the security enforcement mechanisms within the target environment using the security sub-system and language of the host platform. The policy is now able to, in the words of Daminaou *et al*, "... govern the choices of behaviour in a system ..." [27].

Fit for purpose security demands a policy that is able to respond to changing conditions in the environment and effect appropriate behaviour in the enforcement mechanisms within the system. However, a crucial part of balancing the desired levels of flexibility against assurance is exposing the impact and ultimate risk that introducing flexibility will undoubtedly bring so that mitigating action can be taken. The risk assessment process can be assisted, or hindered, depending on the quality of information resulting from the policy modelling process. It is therefore crucial to capture and express security requirements as accurately as possible.

For the purposes of this research, the main body of the investigation is focussed on the use of graphical models to express flexible security requirements. Our work embraces the philosophy of DBSy and judges that the security requirements of an organisation can be captured in models which identify separation and sharing needs. However, in accordance with our vision for fit for purpose security, the DBSy technique is applied to a case study which exhibits changing, temporary and conditional requirements. This will be a typical scenario for the NEC military operation of the future. Since DBSy was developed originally for the defence community, our study provides an ideal opportunity to assess DBSy against a very relevant type of case and identify any limitations with the approach.

During the course of the research it was revealed that separation and sharing requirements could be captured in more than one way using DBSy. This led us to conclude that there was probably no right or wrong approach; the onus is on the system designer to select the approach which most accurately reflects the requirements of the system. This is an important point because critics of our work could suggest other ways in which the case study could be modelled using existing DBSy notation which would of course affect the structure of the model and may impact other factors such as the risk assessment. Given that our concerns with DBSy lie in its ability to capture changing requirements, it is considered necessary to study as many permutations of the models as possible in order to ascertain if change can be reflected by a different method. However, as the work in this section confirms, regardless of alternative solutions, there are limitations with the current modelling approach with respect to capturing change. The solutions we put forward to augment DBSy have been carefully considered; indeed, as the study progressed we discovered ways in which our own proposed additions could be enhanced to address the challenges of presenting temporary conditions in the models.

During the course of this work it was discovered that an important assumption was being made about the user(s) affected by the temporary change in capability. The scenario only affects one user at any one time but the domain structure suggests that all members (which may be one or many) are impacted. The case arises where capability between members of a domain is temporarily unequal which in turn, affects the dynamics and relationships inside of the domain. Although it is possible for the members to operate in different domains and model this case using existing notation, it is believed that such an approach does not accurately reflect the requirements. If the inequality is both temporary and conditional the additional domain may never be required and therefore may never need the security mechanisms employed to support it. To overcome this inconsistency, another concept and addition to the notation was required which could either work with or independently of our NPC2. The result was the Non-Persistent Domain (NPD). The purpose of the NPD is to represent a temporary community created spontaneously (and perhaps conditionally) for a specific purpose. Not only does this concept fit with our philosophy of capturing changing security requirements it could also be used to capture and express the presence of ad-hoc communities and include them in the InfoSec models. To the best

of our knowledge this has not been done with DBSy. As discussed in Chapter 1, the challenge of extending the security boundaries of a system to include such temporary connections is a natural consequence of the Information Age and is increasing in prevalence and importance. It is therefore deemed essential to discuss this development in our work as part of the modelling section.

The chapter begins with a brief overview of DBSy and then presents the Information Security (InfoSec) models which are the output of this modelling process. After the results are analysed, the InfoSec models are updated to include the NPC2 and a summary is provided describing how these additions augment the existing methodology. We conclude the chapter by summarising some of the collaborative work that has been conducted with De Montfort University using their formal policy modelling and analysis tools.

3.1 Example Scenario

In order to validate our approach, a small but illustrative scenario, typical of a future military requirement has been devised. One of the most striking characteristics is that a mixture of resources from different security domains are used to form a new agile grouping suited to the task in hand. In the lifetime of this task it is expected that the boundaries of the system and the separation and sharing requirements will change to reflect operational need. A high-level statement of requirements is given below:-

Military Need: *On an expeditionary operation, the deployed order of battle (ORBAT) comprises groups of military assets that exist within pre-defined security domains. NEC envisages that commanders may create any desired subset as an agile group for a limited period, to undertake a specific task. The elements of the agile group will need to intercommunicate and will also need to communicate with their functional parent organisations. On completion of the task, either the agile group will dissolve or its composition will be adjusted for another task. The commander's ability to task organise an agile group should not be constrained by security considerations, in terms of what can be task-organised, the free passage of information between agile group elements or the time taken to establish the group.*

3.2 Summary of Requirements

For the purposes of this research a formal requirements analysis has not been conducted on the scenario. The above description given of the 'Military Need' provides sufficient information to allow us to elicit and summarise explicit and implicit requirements from the statements of intent. Some of those listed below are used in the development of the models because they provide information regarding separation and sharing needs. The results will show that not all information pertaining to the requirements can be captured and must be therefore be provided in supporting documentation or by some other means. Indeed, the DBSy models are generally accompanied by a rich document set, including tables and possibly network diagrams that assist with communicating the security requirements of the system under development.

Explicit requirements:

E1. It must be possible to create new agile resource groupings from a subset of resources originating from different security domains

E2. Within the new agile group, intercommunication between resources is required

E3. Communication back to the functional parent organisation is required

E4. On completion of the task, assets must be sanitised so data, for example, is appropriate for the security classification of either the functional parent to which the asset is returned or a new group if re-organisation takes place

E5. The Commander requires liberty to organise assets without being constrained by security

Implicit requirements:

I1. Assets from different security classifications and security clearance must coexist in the agile group

I2. The needs of the agile group's commander must not compromise the security of the organisations surrounding the new grouping. Therefore it is implicit that the commander understands the external security environment and the constraints it may place on his or her decision making

I3. The formation of different resource configurations will be time-critical, that is, there will be a time frame outside of which the resources will be of limited or no use (and may even compromise security). Flexibility with the underlying policy and infrastructure must be realised within a specified time period.

3.3 Modelling Security Requirements with DBSy

This section begins with a brief description of the DBSy approach and notation. For brevity this section is limited to an overview. Further information and worked examples can be found in publications such as [16], [17]. Appendix A contains an extract from the DBSy 'quick reference guide', reproduced by permission of QinetiQ.

DBSy captures and documents the sharing and separation requirements of an organisation in a series of Information Security (InfoSec) models. The process begins by defining the InfoSec Business model which serves to identify:

- The resources to separate
- From whom separation is necessary
- The information/resources to share in order to meet business needs
- With whom sharing is permitted to meet business needs
- The method by which sharing is achieved

Resources are grouped together into domains which are analogous to departments in an organisation. A domain has no geographical dependency, that is, it could represent

a department whose members are dispersed physically across the globe. Domain members are at liberty to share information within the domain without constraint. However, the fact that members are grouped together is indicative that the domain requires protection from other domains identified in the system and some constraints are placed on the ability to share. Where sharing is permitted between domains a line is drawn connecting them together and annotated to show the method of sharing.

The InfoSec Infrastructure model builds upon the business model to show where 'strong' separation is required between domains through an impenetrable boundary. Where sharing is permitted, it is controlled through a single, managed point of access. As its name suggests, the Infrastructure model is more closely aligned with the underlying network and controls that would be enforced to implement the separation and sharing requirements that form the security policy.

Finally the InfoSec Architecture model is the composite of both models. This allows stakeholders to combine both separation and sharing requirements onto a single model. The model assists with understanding risks posed to the business which arise from the permission and capability to share data or indeed from preventing it. At this stage of the process the models are completely technology independent. It is only when the risk assessment is performed that system architects calculate the type and placement of controls necessary to reduce risk to an acceptable level.

3.3.1 Modelling the Scenario using DBSy

The scenario, introduced in Section 3.2, has been modelled in Figure 1 using DBSy. Not all of the requirements listed in Section 3.2 can be shown in the models because some would be satisfied by procedural rather than technological means.

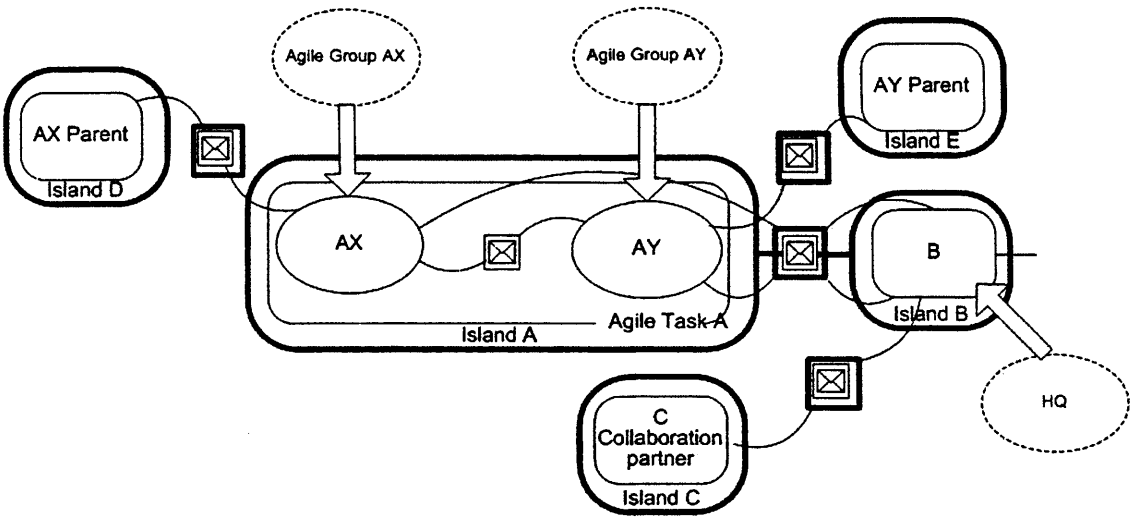


Figure 1: Case Study modelled as InfoSec Architecture Model

Figure 1 shows the InfoSec Architecture model. To aid comprehension of the model, the symbols provided in Appendix A may be used in conjunction with the narrative that follows.

Domains are shown by solid ellipses. Permitted business connections link domains together and are annotated with a construct to indicate the method of sharing. For the purposes of our research it is necessary only to indicate that sharing is required, not necessarily the means of achieving it although Figure 1 assumes 'messaging' as shown by the envelope construct. Environments are essentially 'where people work' and are indicated by broken ellipses. These relate to geographical locations. The portal or broad arrow connecting the Environment with the Domain represents the means of access. This would provide details used for implementation such as two-factor authentication, operating system, access control policy and so on. Figure 1 shows Environment and Portal constructs for the agile mission group (Agile Task A) and Domain B (the HQ). This part of the system is our focus of interest and the elements over which control has been assumed. Therefore we have not identified Environments and Portals for the other domains in the model.

Emboldened rectangles are named 'Islands' and belong to the InfoSec Architecture notation. These are used to show where strong separation, assumed to be

impenetrable, would be required. It can be seen that the Agile Task A Domain containing Domains AX and AY would be hosted on its own island of infrastructure. Sharing between the domains is controlled at a single point, represented by a Causeway or emboldened square which also contains the notation for the method of sharing.

In Figure 1, the Domain 'Agile Task A' has been refined into two sub-domains AX and AY. This allows us to reveal more detail about the composition of the domain and is necessary because there are restrictions placed on the sub-domains which could not be adequately captured in a higher level view. In our scenario, Domains AX and AY experience different external connections albeit under particular conditions. It was therefore necessary to refine the Domain 'Agile Task A' in order to capture this level of detail.

Figure 1 shows the model of the complete case study including the functional parent components of the organisation. Furthermore, an element of prediction has been included in the models by introducing an external domain to represent collaborative partners. The case study states that NEC envisages that commanders may create any desired subset as an agile group for a limited period. Given that there is an increasing requirement to share information within national contingents and between coalition partners, it is reasonable to include an external domain to the model to indicate with whom collaboration may become necessary depending on how the operation unfolds. It should be noted that this requirement demonstrates a potential flexible and conditional need in that it *may* happen. It is by no means certain. The work in this chapter will reveal that it is possible to capture a conditional requirement in the graphical models and in Chapter 4 the potential impact on risk can be assessed.

Our scenario has an additional consideration, not captured explicitly in the scenario text, but which gives rise to the need for flexible and 'adaptive' behaviour and therefore serves to illustrate the *raison d'être* of this research. The condition affects the agile group and we judge it to be a reasonable and realistic requirement for a military hierarchy. Under normal circumstances Domain AX is permitted to share information with Domain AY. It is not normally permitted to share with Domain B *unless* Domain AY is unavailable. This requirement is fundamental to our research

since it exposes the need to capture changing capabilities within the graphical models. This type of behaviour would undoubtedly occur as a task develops because, as indicated in [63] it is expected that a plan never survives first contact and information needs would change to support modifications to the plan. Moreover it is probable that the configuration of a system would change with respect to ad-hoc interactions between different domains. Small communities of users with shared interests would be set-up and dismantled for the purposes of temporary information exchange.

The InfoSec Business model maps to the requirements listed in Table 1 as follows:

Requirement	**Achieved in Business Model**	**Comments**
E1	No	Not possible in model
E2	Yes	Agile group members AX and AY permitted to share
E3	Yes	e.g. Domain AX permitted to communicate with AX Parent domain
E4	No	Not possible in model
E5	Achieved in part	1. Model contains element of future prediction with inclusion of external collaborative domain. 2. Connection shown between domains AX and B. This relationship will be clarified since the connection is only permitted under particular conditions.
D1	Achieved in part	DBSy notation allows us to refine a domain into constituent parts. In this case AX and AY have different levels of capability
D2	No	Not possible in model
D3	No	Not possible in model

Table 1: Requirements Mapping

The results in Table 1 show that the DBSy models are suited to capturing certain requirements; specifically those cases where there are clear and unchanging separation and sharing requirements between different communities of users. Other requirements they can only capture in part or not at all. Our work looks in detail at some of the requirements that are partially satisfied. It is surmised that this is where

the research will provide the greatest contribution and progression towards the goal of fit for purpose security.

The investigation continues by refining the scenario to concentrate on a sub-set of the system, the Agile Task group which exhibits the changing security properties. The resulting InfoSec Architecture Model is shown in Figure 2.

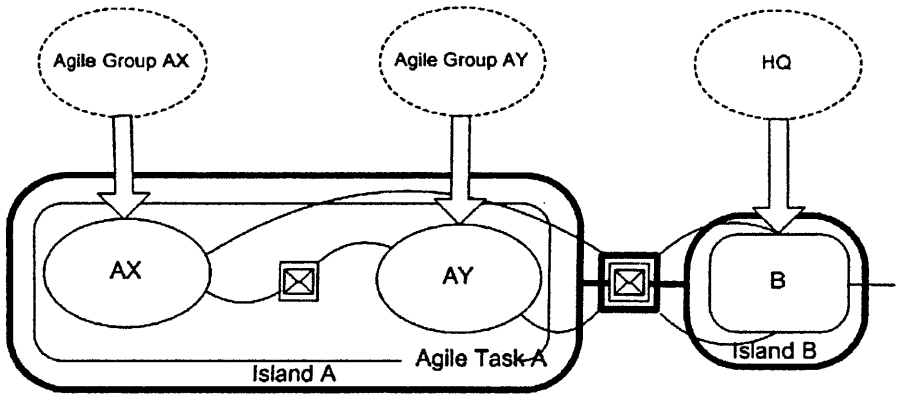


Figure 2: Focus of Interest - Agile Task Group

As shown in Figure 2, DBSy captures maximum permitted connectivity, that is, all of the permitted connections that would be required within the system at any time are represented in the models. One very important observation of both Figures 1 and 2 is that permitted connections between Domains AX and AY and AX and B are both shown. It is deduced from the requirements that this observation is not strictly true. In this case, the ability to share is a *conditional* capability and will therefore only occur in specific circumstances. In a dynamic environment maximum permitted connectivity is context specific. It would be expected to change as a situation or operation unfolds to reflect changing (permitted) resource requirements. For example, our policy may state that information up to secret can be shared within the community. However, if a collaborative partner is encountered and communication established, this may change. How can this situation be captured in existing approaches, that is, do we model the maximum state of connectivity where secret information is being passed? If this approach is adopted, how do we manage and model the situation where a collaborative partner is encountered and a different level of information sharing is permitted? Furthermore, it is known from the scenario that sharing is only permitted between Domains AX and B when particular conditions are

present in the network. The models in Figures 1 and 2 only capture the maximum state, which is in fact, inaccurate. How can we reconcile the two sets of capabilities with one another? These questions are addressed as part of this section.

In order to have a comprehensive understanding of the security landscape and the appropriate security enforcement controls in the system it is necessary to express what we term 'non-persistent' business connections. These temporary and possibly conditional connections between domains essentially give the system its flexibility since they represent the requirement to share (resources) between domains but temporarily and on a conditional basis. Quite simply, they alter the separation and sharing profile of the system under different circumstances. With access to this level of information, the system architect is able to select appropriate security controls, when required, to enable a capability. The models should provide sufficient detail to aid understanding of the impact that enabling flexible behaviour will bring. This view is entirely complementary to and supportive of our notion of fit for purpose security.

Where a system is likely to be deployed in different scenarios, it is expected that the system would be modelled for each individual case rather than attempt to capture all of the information on a single model. The changes of state, represented as non-persistent connections within this case study, could be considered as individual scenarios. Therefore individual models have been produced in Figures 3 and 4 to capture the two states of the system. We term this *level one refinement.* Under this regime, not only is it possible to capture the true connections at different states in the policy, we can also identify limitations with the models more easily and demonstrate where our proposed additions augment the modelling process.

In accordance with our case study, Figure 3 shows the business model where connectivity is permitted between Domains AX and AY but not between Domains AX and B. This state is categorised as the baseline policy since it represents the normal set of conditions that this system is expected to support.

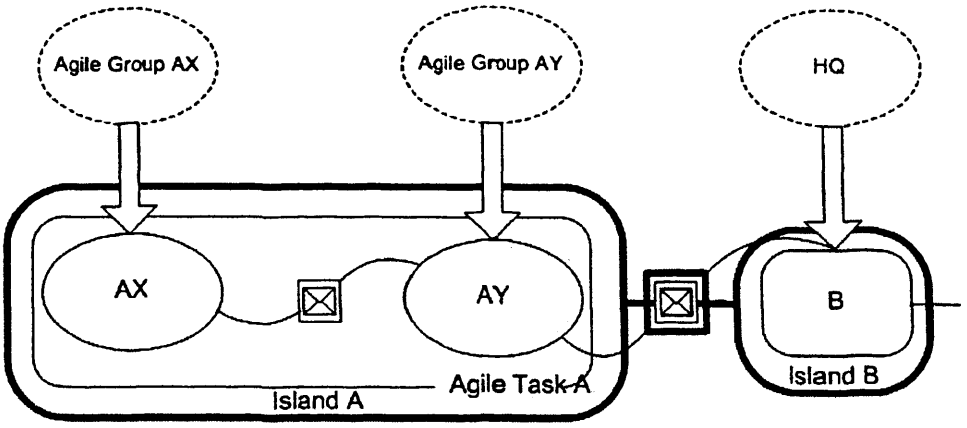


Figure 3: InfoSec Architecture model; Domains AX and AY sharing

Although Figure 3 is an accurate reflection of the 'normal' policy state it does not reveal exceptions to the rule. It would therefore be necessary to refer back to a more detailed diagram in order to fully appreciate the complete picture of connectivity.

Figure 4 shows the exceptional state of the policy where Domain AX is permitted to communicate with Domain B but not AY. Since this model is a refined version of Figure 2, none of the related connections are shown; specifically the relationship between Domains AX and AY is not provided.

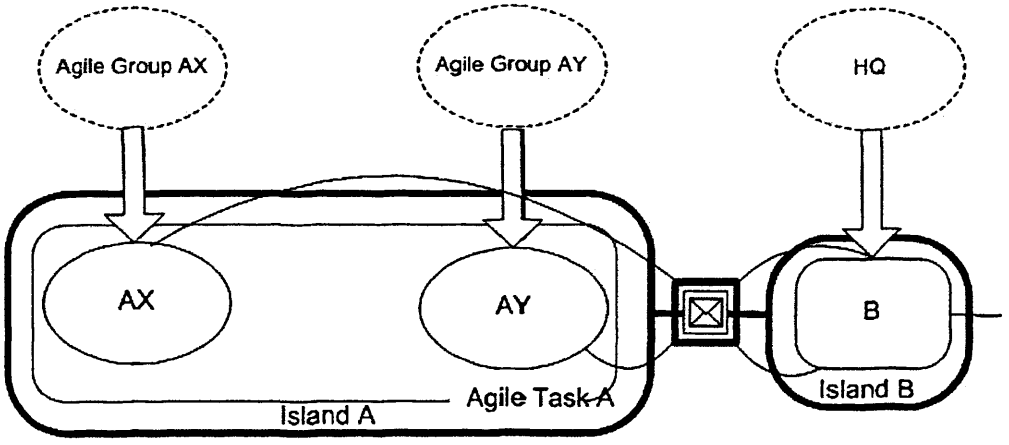


Figure 4: InfoSec Architecture Model; Domains AX and B sharing

3.3.2 Limitations with DBSy

The main limitation revealed through the modelling process so far is that the DBSy models have no cognisance of temporal and conditional changes to the separation and

sharing profile. The security solution is always based upon maximum permitted connectivity irrespective of the fact that sharing may never be required between some of the domains if the motivating condition does not arise. Currently, different 'scenarios' are modelled independently of one another therefore each time a condition alters the state of the security profile, it would be necessary to generate a new model. We conclude that if conditional and temporary connectivity can be represented on a single model, the fidelity of the models is enriched. The information presented to system designers and stakeholders is a more complete and accurate representation of the actual security requirements.

3.3.3 Application of the Non-Persistent Capability Concept

Our research proposes a simple but influential addition to DBSy that will identify where a temporary and conditional business connection between two domains is permitted. The construct put forward in this thesis represents the NPC2. Its purpose is to alter the separation and sharing profile of the system by manipulating the presence of the business connection. Note that the connection indicates permission to share which of course, requires a number of complementary activities to occur before this can take place. In [23, p.125] we describe how the activity is decomposed into a series of stakeholders, interactions and attributes that allow us to express 'permitted to share' in terms of policy rules. However, at the level of abstraction required for this discussion it is sufficient to simply capture the existence of the behaviour through the NPC2 construct.

Our additions to the DBSy notation include two constructs which operate together. The first is a conditional connector, which takes the form of a switch and is shown in Figure 5. This depicts the conditional paths through the construct when the input point and an output point are connected by a line.

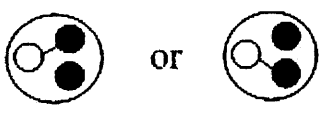


Figure 5: Conditional Connector

The second is a temporary business connector, represented by a broken line (connecting one point on the conditional connector with a Domain). The use of a broken line is deliberate as it allows us to distinguish between the standard permitted business connector, which would form our 'normal' or 'baseline' policy and temporary exceptions to this case. The temporary business connector is shown in Figure 6. Note that it can be used independently of the conditional connector and in Chapter 4 such a case is described.

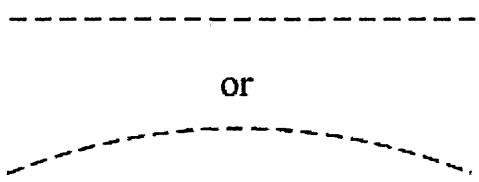


Figure 6: Temporary Business Connector

Combined, these two objects comprise the NPC2. Our proposed modifications to the notation are extremely simple in design but far-reaching in impact. Our decision to maintain simplicity with the proposed additions was deliberate and entirely in harmony with the design principles of DBSy. The NPC2 was put to QinetiQ and the reaction to the notation itself was mixed although the principle behind its purpose met with a positive response. Acceptance of the ideas into the DBSy notation has not been pursued further but will probably be followed up once the thesis is complete.

Figure 7 presents the modified InfoSec Architecture model using the NPC2.

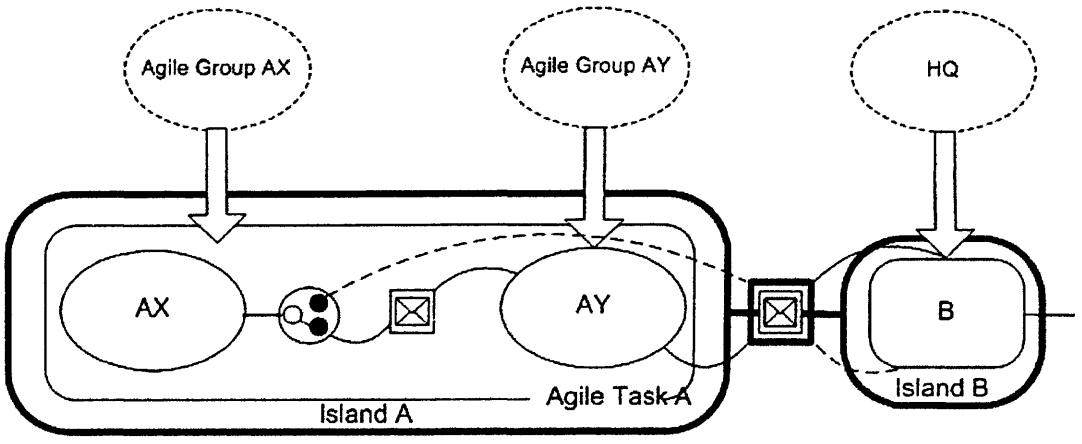


Figure 7: NPC2 in DBSy InfoSec Architecture model

Figure 7 shows that Domain AX is permitted to share with both Domains AY and B, but *not simultaneously*. The current (active) path is shown by the position of the switch. Importantly, the temporary business connector provides information about the policy in that it shows the exception to the 'normal' connectivity. Figure 7 informs us that the standard condition is to permit sharing between Domains AX and AY because it uses the standard DBSy business connector. The exception to the policy is the temporary business connector allowing communication between AX and B albeit under specific conditions.

The NPC2 is clearly useful in clarifying the nature of a relationship between two domains where their separation and sharing profile is not constant. In Chapter 7 additional examples derived from the case study are modelled to show other instances where the NPC2 can be used to differentiate between baseline and exceptional or changing security requirements. Nonetheless, despite the positive advantages of introducing the NPC2 to the models it is still believed that, in some cases, a level of detail is missing from the models.

Use of the domain construct implies that *any* or *all* members may share information with the connected Domain using the permitted business connection. This is not necessarily the case. Our scenario provides an example where the military organisation would follow an established hierarchy of capabilities. Under this regime, only one member (the second in command or 2IC) would be permitted elevated capability if and when the requirement arose. Although it is anticipated that DBSy would require separation of these users into separate domains to reflect their different 'needs', it is suggested this approach is not always suitable. In our scenario for example, the requirement for the 2IC to 'move' between domains would only occur under certain conditions. It is therefore a temporary and exceptional requirement which should, in our view, be documented as such. This approach more accurately states the separation and sharing profile of the system and benefits the correct assessment of risk. Our alternative solution is the NPD, which complements the NPC2, and is described in the next section.

3.3.4 Introducing Non-Persistent Domains

The NPC2 captures the existence of temporary changes to the separation and sharing profile of a system that may arise as a result of different events in the environment. We conjecture this also includes the formation (and destruction) of ad-hoc communities where members will form a temporary association for the purposes of information sharing and disband when appropriate.

As described in the previous section a potential complication arises from the implied assumption that in forming a non-persistent connection between Domains AX and B the models effectively state that *any* member of Domain AX is permitted to communicate with Domain B. In reality, this would not necessarily be the case. In an agile mission group the 2IC would take on the additional responsibility to maintain communication, not the entire membership of the community. In the absence of the 2IC an established hierarchy would be followed where the next most senior would become leader and so on until membership was exhausted! In a DBSy model an individual can be a member of one or more domains and may work in more than one domain at any one time. This access is shown and ultimately controlled by the environment since it is probable that the domain would only be accessible from a particular configuration or possibly geographical location. As part of our research this approach was considered but rejected for the following reason.

It is feasible that a member of Domain AX could also be a member of Domain AY and simply transfer to the alternate domain when conditions in the environment required this behaviour. However, the ability to work in Domain AY may be impractical since it could afford the user inappropriate capabilities for the task in hand. Domain AY could, for example, hold permitted business connections with other domains that should not normally be accessible to or exercised by a 'temporary' member originating from Domain AX. Recall that our aim in fit for purpose security is to support *appropriate* security at the time it is required. We therefore seek to enhance the user's capability for a limited set of tasks that enables them to do a specific task at a specific time. This approach allows them to either remain in their own domain or through the use of our NPD, transfer into a new 'temporary' domain constructed specifically for the purpose. In this section we illustrate how this can be

achieved. For brevity notation representing the Environment and Portal constructs has been omitted from the diagrams.

In Figure 8 Domain AX has a permitted business connection with Domain B as indicated by the position on the conditional connector. Not all members of Domain AX have this capability therefore it is necessary to differentiate between regular members of Domain AX and the member with elevated capability.

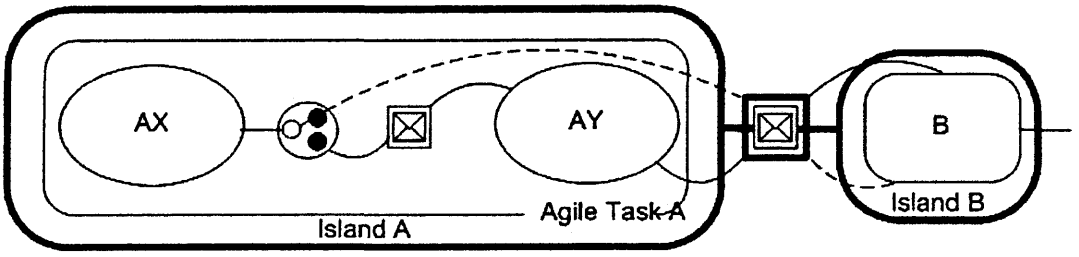


Figure 8: Logical connectivity when NPC2 is employed

Figure 9 illustrates what *actually* happens when the conditional connector permits the business connection between Domains AX and B.

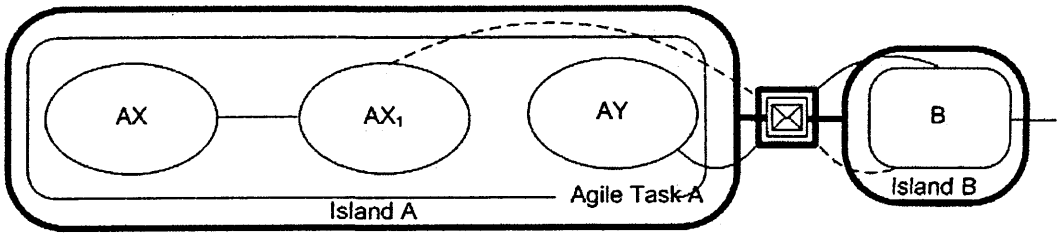


Figure 9: Actual connectivity when NPC2 is employed

The elevated capability associated with the 2IC to permit the business connection with Domain B effectively sets this individual into their own domain (AX_1) since they have software acting on their behalf to allow sharing with Domain B. Other members of Domain AX either do not have this capability or if it is available, it is neither active nor accessible to them. Although the refinements shown in Figure 9 are a better reflection of the case study it is still not completely accurate because the temporal characteristics of the scenario have not been identified. Domain AX_1 may effectively never exist if the condition that triggers the need for it does not transpire.

Closer examination of this case leads us to conclude that an additional construct, which represents a temporary domain, may be the answer. Since the NPD is a temporary community we selected an elaboration of the existing domain construct and devised the ephemeral 'cloud'. At this time neither the concept nor the notation for the NPD has been put to QinetiQ as possible additions to DBSy. Figure 10 revises the InfoSec Architecture model to include the NPD.

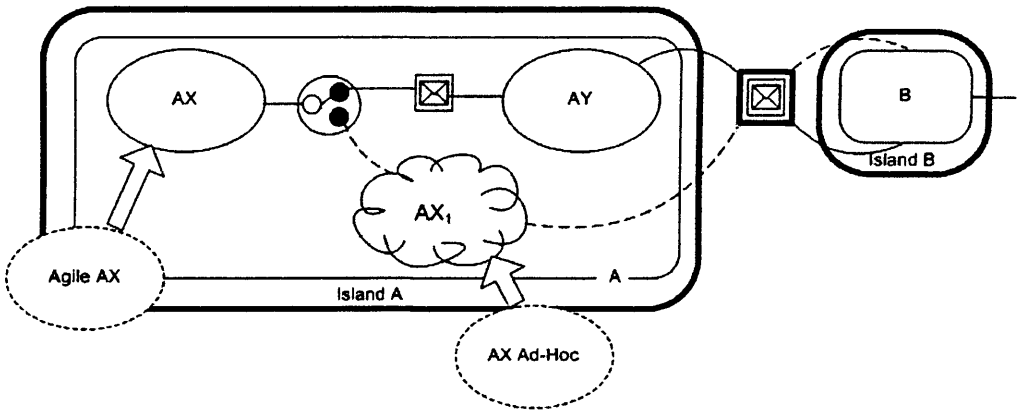


Figure 10: Introducing the NPD

Figure 10 is interpreted as follows. Domain AX is normally permitted to share information with Domain AY as shown by the position of the NPC2. It is not normally permitted to share information with the temporary Domain AX_1 and does not have a direct business connection with Domain B. Domain AX_1 is a non-persistent or temporary domain which, in this case, only exists if the NPC2 is switched to allow a permitted business connection between Domains AX and B. When formed, AX_1 is permitted to share information with both Domains AX and B. As shown in Figure 10, it is part of the Agile Group Domain and Island and is therefore subject to the same security constraints.

Note that Figure 10 also shows the DBSy Environment and Portal constructs. Domain AX may be accessed from the 'Agile AX environment' whilst AX_1, when it exists, may only be accessed from the AX Ad-Hoc environment. The naming of the domains and environments is quite deliberate and attempts to indicate the relationship, in this case, between Domain AX and the non-persistent Domain AX_1.

It is possible to extend the concept of the NPD by looking at other types of domain occupant. For example, Domain AX_1 has only been constructed to permit sharing with Domain B. However, it can also be used to explain the case where a domain is (potentially) permitted to share with a collaborative partner. An ad-hoc domain using the NPD construct could be employed for this purpose as shown in the InfoSec Business Model in Figure 11.

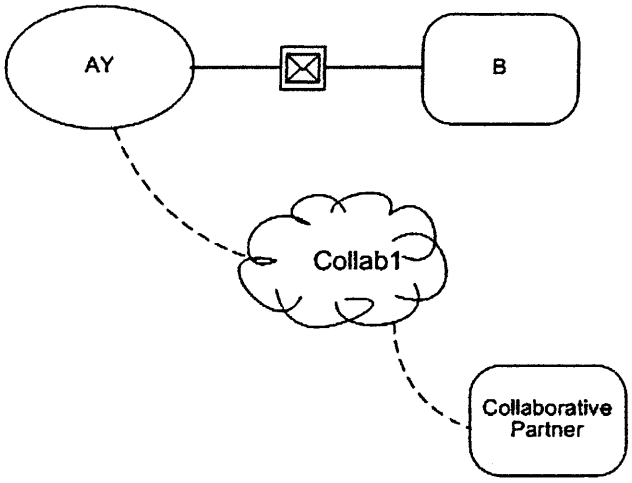


Figure 11: NPD representing an Ad-Hoc Community

In Table 1, requirement E5 stated the Commander's need to (re)organise assets without constraints from security. For the purposes of illustration, the requirement was interpreted in such a way that a (potential) permitted business connection with a collaborative partner should be shown in the model. This provides an example of a future requirement which is not definite at the time of planning, but allows us to predict what could happen if the agile mission group did need to organise in such a way as to share information with a third-party community. Whereas the original DBSy model could only partially capture the requirement, it is believed that the NPD used in Figure 11 fully captures it. It shows that Domain AY may form a temporary permitted business connection with an ad-hoc community which is also occupied by the collaborative partner. This configuration may have additional constraints placed upon it (not shown in this Figure). Since the NPD represents an ad-hoc network, it is quite possible that other members apart from the collaborative domain which has been anticipated may be present. For that reason the NPC2 may also be used to show that

when the NPD is established, Domain AY may not communicate with Domain B. This InfoSec Business model for this scenario is shown in Figure 12.

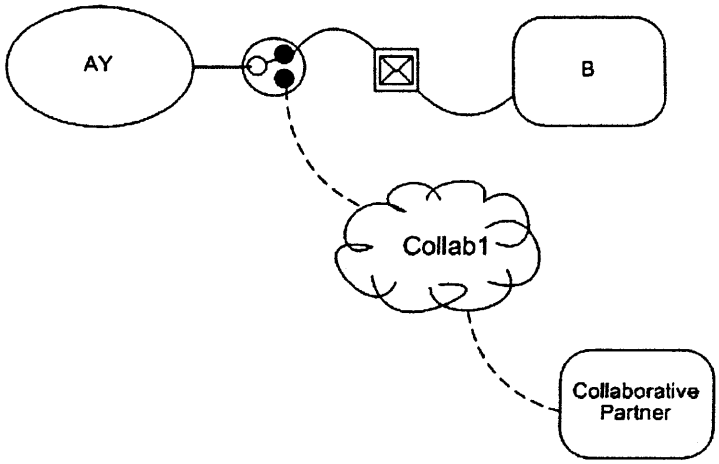


Figure 12: Combined use of NPC2 and NPD

Figure 12 shows quite clearly that the baseline policy permits Domain AY to share with Domain B using messaging as indicated by the NPC2. When switched between Domain AY and the NPD, the business connection with Domain B would no longer be permitted or available as shown in Figure 13.

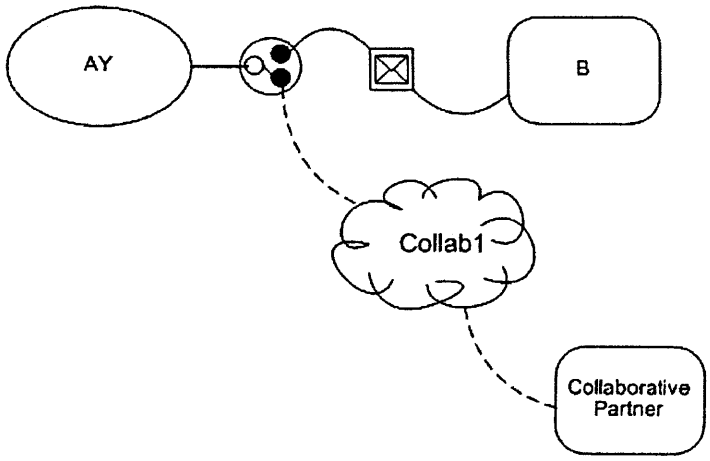


Figure 13: NPC2 permitting business connection with NPD

This section on informal modelling can be summarised as follows. Capturing and expressing temporal and conditional separation and sharing requirements in a high level graphical modelling environment is difficult, particularly when working within the boundaries of the existing toolset and avoiding radical changes to either the

approach or notation. Our proposed additions to the DBSy methodology and notation are promising. The NPC2 allows us to capture the existence of a non-persistent permitted business connection between two domains. However, we note that its use suggests that an entire domain adopts any change in permitted connections when it is likely that this capability may be limited. The fit for purpose security approach seeks to minimise the number of changes to the security profile and elevate capability for a limited period thus affecting the minimal number of users possible. This approach is unlike others, such as the principle of weak tranquillity, where the protective marking of an object is lowered (after sanitisation) with the potential that a greater number of subjects will subsequently acquire access.

There are of course different ways in which the requirements could be modelled. However, without a change in the notation, it is not possible to capture the temporal or conditional properties of a connection. Through the introduction of another construct, the NPD, ad-hoc groups of subjects can be identified. As shown in Figure 10 by using the combination of the NPC2 and the NPD construct it is possible to identify a temporary change in the separation and sharing requirements, including the existence of a temporary community. Of course the level of detail is limited. The use of the NPC2 shows a relationship between the change in separation and sharing and the use of the temporary domain however, it does not inform us if the domain is formed as a result of the change or if it pre-exists. From the use of the environment construct we learn that access to the temporary domain is achieved through the AX ad-hoc environment therefore if a subject requires access to both, each environment would need to be supported.

It is acknowledged that the additional constructs introduce some complexity into the model and there are potential inconsistencies that need to be addressed. For example, the connection between AX_1 and B has been shown as temporary in Figure 10, that is, a dependency exists on the conditional connector and the associated permitted business connector. If the conditional connector permits sharing between AX and AY then AX_1 effectively does not exist because there is no requirement for it. However, we could also utilise the NPD construct to represent an ad-hoc network. In this case its existence may not be dependent on the availability or not of Domain AY. Furthermore, the creation of an ad-hoc network may be in addition to or in place of an

existing domain. Non-persistent constructs and ad-hoc networking is an area of research in its own right. It is considered important to model a simple case study to begin discussion, and this is addressed in Chapter 7. Even so, we do not claim to have exhausted discussion of this topic in this thesis, and it is clear that further research could be done.

3.4 Formal Policy Models

So far this chapter has described the role of the NPC2 and NPD within the context of an informal modelling approach. Given that our work with fit for purpose security combines both flexibility *and* a level of assurance it is deemed necessary to consider an informal methodology such as DBSy in conjunction with a formal modelling approach based on mathematical proof. This approach supports a more rigorous level of analysis and verification to confirm the extent to which the proposed system and policy design conforms to the security requirements. This is clearly only one level of verification; once the design is translated into a working system it is necessary to conduct a thorough set of functionality and penetration tests to confirm the security and functional behaviour of the system under both 'normal' and 'exceptional' operating conditions.

This section introduces the collaborative work that was conducted with the Software Technology Research Laboratory (STRL) at De Montfort University. The STRL has been actively engaged in security research particularly in the area of software Agents and SANTA is one technology to emerge from ongoing research [50]. The collaboration provided an opportunity to use the DBSy InfoSec Architecture models from our case study as input to the SANTA formal policy modelling process and analyse the success of combining the two. This work had several aspirations. Not only were we interested to discover whether the dynamic requirements described in the case study could be modelled using SANTA, we were also curious to find out how well the output from an informal approach could be used as input to a formal approach that defines and requires very precise constructs. The study for this thesis benefited from verifying the changing and conditional separation and sharing requirements expressed in DBSy through a formal modelling approach and the STRL benefited from exercising SANTA with a case study from a different domain.

Furthermore, it was possible to create a simulation of the security policy using SPAT which is a part of the toolkit developed at the STRL. The simulation provided verification and feedback through a graphical interface confirming that the policy rules were behaving as defined and expected as the conditions changed. For the purposes of this thesis, detailed elaboration of the collaborative task is not appropriate. Instead we summarise some of the salient content and results from detailed papers as published in [23] and [24].

3.4.1 Policy Modelling

The case study used in [22] was essentially the same as that described in Section 3.3.1. The paper referred to the limitations of DBSy regarding the capture of dynamic security requirements within the existing notation. Our work with the NPC2 was in its infancy at that time. Therefore, the role of a non-persistent capability concept such as the NPC2 was only mentioned but not presented in the models.

The SANTA security policy framework comprises three main components. Firstly, the policy model allows for the expression of security requirements such as:

- *Authorisation*: Defines access to resources such as the communication infrastructure
- *Delegation*: Defines which stakeholders can pass on their privileges to others
- *Obligation*: Defines conditions under which activities must be performed

The model has a formal underpinning in Interval Temporal Logic (ITL) [51] which is suitable for expressing and reasoning about temporal dependencies between policies and the dynamic change of these requirements in a compositional manner. By this it is meant that different parts of a policy may be specified individually and then composed to provide the complete system policy.

The formal foundation and the expressiveness of the policy model with respect to dynamic change and temporal dependencies of security requirements was used to show how high-level DBSy specifications can be represented by policies. Classes of requirements that cannot be adequately represented using DBSy were identified and modelled. The approach described in the paper exploits the advantages of both

methodologies: the intuitive and high-level modelling capabilities of DBSy, and the expressiveness and formality of SANTA. This complementary approach should appeal to a wider policy development community and ultimately increase the opportunities for developing fit for purpose security.

For the purposes of the paper, we extracted a number of policy requirements from the DBSy model that enabled us to illustrate our interest. The study was successful. The requirements were modelled using SANTA and the resulting policy was validated with SPAT. Not unexpectedly, the process did reveal difficulties when correlating the two very different modelling approaches. For example, before security requirements can be refined all stakeholders and their potential interactions must be identified. In this case study stakeholders were of course the members of the different domains, thus a domain represents a community able to interact. The stakeholders can be identified from the DBSy model with relative ease. Establishing the potential interactions is more difficult, as the model can only express the existence (or non-existence through omission) of a business connection. That said other information can easily be identified including the mode of sharing, which was messaging in this case study. Moreover it can be inferred from a two-way business connection that stakeholders are permitted to both 'send' and 'receive' information across the permitted connection. The domains themselves are usually (but not always) labelled with the maximum protective marking of the material that can be contained in the domain. We therefore know the range of sensitivities that may be transmitted. What the models do not reveal however, is any constraints or conditions that may be placed on the interaction. A policy language, such as SANTA, provides greater expressiveness when defining the constraints on a business connection. The condition under which the communication is permitted may be more complex than domain membership. Policies can also explicitly state conditions for a *denial* of an interaction, which cannot be expressed in the higher-level DBSy diagram. To resolve conflicts, that is, where both a permission and a denial can be derived from the policy, the policy contains decision rules that enable conflict resolution.

3.4.2 Conclusions from Collaborative Study

The joint work with De Montfort University was a useful exercise because it demonstrated how the graphical and visually intuitive DBSy models could be translated into a more expressive policy language such as SANTA. The formalisation process allowed for the identification and correction of ambiguities in the high-level policy that would not have been detectable otherwise. Given the policy representation of the DBSy model, it was possible to revisit the original requirements and capture more sophisticated properties associated with individual business connections that are of course fundamental in producing the final policy enforcement rules.

The collaborative study demonstrated that flexibility can be achieved through policy and furthermore the use of a formal language provides a level of proof and confidence that is required in a military system. As a result, it is concluded that the approach constitutes an initial step in developing systems that can achieve the military requirements of flexibility and high assurance.

3.4.3 Concluding Observations

One of the reasons for collaborating with De Montfort University for this study was that SANTA is still relatively new. At the time of writing, it was not at a suitable state in its development to allow a novice user to take the software and model a case study without additional support. A watching brief is being maintained to see how the work continues to develop because as the results of the joint paper clearly show, there is an opportunity to use the framework to provide a formal underpinning and refinement of informal models. A promising direction for future work could include combining a formal approach such as SANTA with an informal and graphical approach such as DBSy into a single package. Our collaborative work leads us to conclude that this development could have appeal because it builds upon the intuitive and graphical characteristics of DBSy by offering a level of rigour that may only be achieved through modelling frameworks underpinned by a formal, mathematical approach. This could be a fruitful topic for future research.

Chapter 4: Risk Assessment and Changing Security Requirements

In Chapter 3 the security requirements of the case study were presented in the context of a separation and sharing profile and articulated through the DBSy InfoSec models. This chapter explores the risk analysis process and broadly applies principles from DBSy to our models both with and without the NPC2 and NPD notations. This enables us to validate our approach and assess the impact of including non-persistent and conditional connectivity in the models.

Permitted business connections between domains, which are necessary to enable sharing, also provide the means for compromising security if not appropriately protected. They can therefore also be viewed as *potential compromise paths*. As stated in Section 1.4.1, DBSy identifies a number of possible different threats to the system which would normally be included as part of the risk assessment process. However, in order to assess the impact of the NPC2 and NPD in the models, it is sufficient to limit our investigation to the fact that a connection and therefore a potential compromise path exists, rather than the multiplicity of methods by which it could be compromised. Assuming that the NPC2 and NPD constructs serve to reduce the number of potential compromise paths, it is conjectured that increasing the complexity of the model by considering other factors will amplify these beneficial effects. The DBSy risk analysis process looks at the affect of one or more domains, which form the potential Attack Group (AG), on another, which is the Focus of Interest (FOI). Essentially the FOI is where security controls will be applied to address the methods of compromise which have been identified. It therefore represents the domain over which the system stakeholders can exert some control. The risk analysis process considers both inbound and outbound paths between the AG and FOI. An inbound path *introduces* a compromise to the FOI. A computer virus is one example. The outbound path may constitute a breach of confidentiality such as inappropriate release of information from the FOI. Where the FOI and AG have been defined in our study only inbound connections are included in the calculations. This is sufficient to examine and demonstrate our work. Outbound connectivity would simply double the number of potential compromise paths since the inbound connection could also be used as a means of compromising information from the FOI. This thesis also explores the case

where *any* connection in the model may be considered as a potential compromise path. Whilst the identification of the FOI and AG allow us to define and limit the scope for analysis, we believe it is also of interest to examine the proliferation of connections and potential compromise paths in a system to identify any useful patterns that could be used to assist in assessing risk.

Our work also considers higher order connectivity where *indirect* compromise paths occur in the models as a result of permitted business connections between domains. In order to keep the investigation to a manageable level, we calculate indirect connectivity to a level of two hops between domains. As shown in this chapter and again in Chapter 7, useful results about the number and position of connections can be identified with this level of analysis. This can be used to inform us about the potential accessibility and therefore vulnerability of a domain. To assist with calculating higher order connectivity this research has adopted the formal technique of matrix algebra to analyse connectivity between domains. The matrices lend themselves to computer programming which has the benefit of speed and accuracy in the calculations even when the system is scaled in terms of size and complexity. It is possible to extend our calculations to 3 hops or more, but this is not necessary to illustrate our work.

In this thesis the research adopts a different view of compromise paths from DBSy which is more in line with our vision of fit for purpose security. DBSy works on the notion of maximum permitted connectivity where there is no differentiation in terms of the compromise path between a *potential* connection and an *actual* connection. By contrast, our work recognises changes in the separation and sharing profile and using the NPC2 and/or NPD it identifies where a temporary and possibly conditional exception to the baseline profile may exist. This approach allows us to calculate permitted business connections when either the default policy is active or any of the exceptions are active. It is our belief that this presents a more accurate representation of the actual compromise paths and potential risk to the system at that time.

This chapter begins by modelling risks associated with the agile mission group and considers how temporary and conditional connections such as those modelled with the NPC2 and NPD can be included in the risk calculations. This is followed by some preliminary investigations into calculating the growth of potential compromise paths

as domains are added to the network. Here we are specifically interested in the work of Metcalfe [45] although there are some important differences between Metcalfe's work and our own. Equation (1) is the expression for Metcalfe's law [45] for a fully connected network.

$$N = n(n-1)/2 \qquad \ldots (1)$$

In (1) n refers to the number of nodes in the network and N the total number of unidirectional connections. The expression ignores bi-directional connections (hence the division by 2) and the case where a domain is connected to itself (hence $n-1$). Unlike Metcalfe our work does not assume that the network is fully connected all of the time. Furthermore, because a connection could be used to compromise the domain at either end we do consider bi-directional connectivity. Therefore our version of Metcalfe's law is $N = n(n-1)$. The NPC2 and NPD vary the level of connectivity in accordance with a security policy and we hypothesise, will impact the rate of growth as domains are added to the network. Another important difference lies in our view of connectivity which regards all connections as potential compromise paths. The *value* of a connection is viewed more in terms of its benefit to the unauthorised user than value to the rightful owner. This is an important point. If some domains are more connected than others then it follows that they may also have more value to the prospective hacker for use as a platform from which to launch an attack. The richly connected domain may itself hold low classification data but provide an ideal vehicle to access a more sparsely connected but sensitive domain. The security analyst requires an expedient and accurate method of revealing information about connectivity and growth before it is possible to use this information to design the security of the system under *all* of its conditions. It is our intention to show how this problem can be approached using the NPC2 and NPD in slightly larger configurations which indicate how connectivity is affected by the addition of domains to the network.

4.1 Informal Validation using DBSy

Risk assessment is a complex process involving a number of different techniques that will enable stakeholders to understand and mitigate identified risks to the system. An

InfoSec Architecture model-based risk analysis is one aspect of the overall assessment process. It involves analysing the infrastructure and business connectivity to identify where vulnerabilities may arise in the system through the identification of potential threat actors and agents [18], [19]. The analyst must consider unintentional or accidental compromise in addition to deliberate action with a malicious intent since appropriate security controls must be capable of mitigating both types of activity.

In this research the risk analysis process is based on the InfoSec Architecture model for the Agile Mission group in Figure 2, which for convenience is repeated in Figure 14. This model identifies all permitted business connections, that is, no attempt has been made to differentiate between temporary and conditional or permanent business connections.

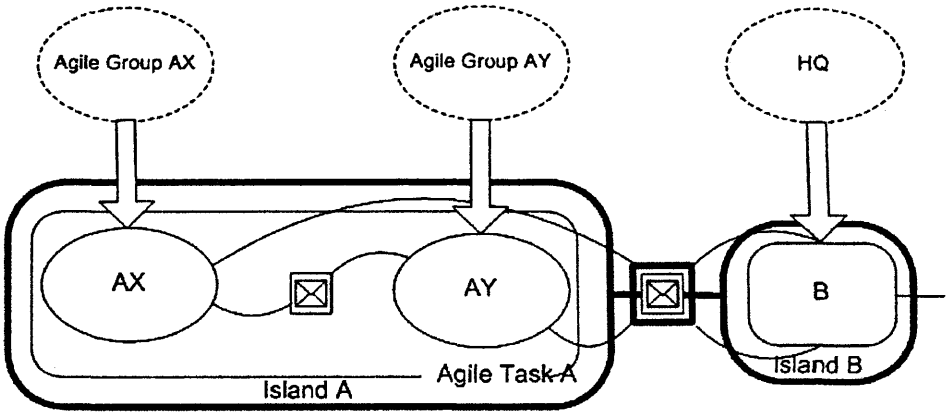


Figure 14: Agile Mission Group showing maximum permitted connectivity

The first stage of the compromise path analysis involves redrawing the InfoSec Architecture model where the domains and underlying infrastructure are separated as shown in Figure 15. For ease of analysis the FOI and AG are both identified. In this example Domain B, the FOI, should be protected from potential compromise by Domains AX and/or AY, which form the AG. The boxes at the top of the diagram represent the domains and those at the bottom, the infrastructure. Lines are drawn representing inbound compromise paths from the AG to the FOI. As previously stated, potential compromise paths that arise as a result of the infrastructure are not included in our initial calculations. They are included in the first model for completeness but omitted from future diagrams.

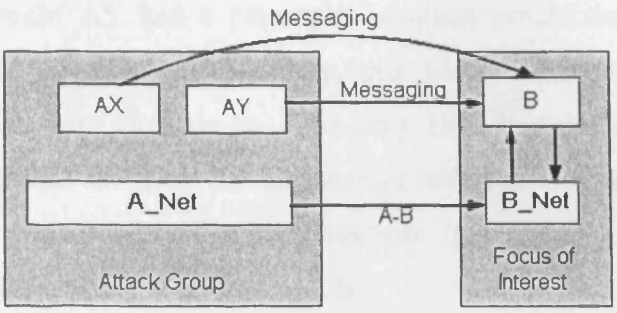


Figure 15: Compromise path model for case study

A couple of points arise from Figure 15 which merit discussion. For example, Domains AX and AY appear as separate entities because quite simply, they are both members of the same attack group. In this context they are not considered a threat to one another despite the fact they are connected because the analysis is considering the impact of the *whole* AG on the FOI. The guidelines outlined in [64] describe the process for simplifying the models which should, in theory allow Domains AX and AY to be combined. However, in our scenario simplification is not practical because each of the two domains in the AG potentially have a relationship with the FOI. If the model is simplified, this level of detail is lost and the resulting model is not an accurate representation of the case study. It is conjectured that in addition to drawing Domains AX and AY it would also be useful to include the connection between the two domains. Although it has no impact on the number of direct or single step compromise paths, it does form an indirect compromise path between Domain AX and B which would become apparent when higher order connectivity is calculated.

Of course, Figure 15 does not and can not capture the conditional nature of the connection between Domains AX and B. Critically for our research, the permitted business connection between AX and B would not exist under normal or baseline conditions. As a result, the only means of compromise between AX and B would arise from indirect connectivity. The above example in Figure 15 shows that two inbound compromise paths exist via the permitted business connections involving Domains AX, AY and B and is therefore erroneous. We conclude that it is necessary to capture the nature of the link between Domains AX, AY and B so that conditional behaviour of connection and potential compromise path can be calculated accordingly. Furthermore Figure 15 does not reveal the relationship between Domains

AX and AY. Domain AY has a permitted business connection with Domain B. However, under the stated security policy, this has no impact on Domain AX's permitted connection with Domain B. The only time this particular connection is allowed and established is when the connection between Domains AX and AY is unavailable. This path is crucial in fact, because it is the feature that 'opens' the potential compromise path between AX and B.

To summarise, the inability to annotate and calculate temporary permitted connections yields misleading results. Only maximum permitted connectivity can be shown. This identifies all possible compromise paths regardless of the fact that some of the paths may not be active in different states and indeed may not exist at all unless the condition prevails which triggers a change in the security profile a temporary permitted business connection may never be required. The standard approach calculates the effect as if it were in use. It is concluded that the process would be more useful if the nature of the connections is captured, because it would then show the maximum number of connections under various states. The system designer can then mitigate those risks accordingly and deliver appropriate security when it is required: in essence, realise 'fit for purpose' security. Of course we believe this can be achieved by including the NPC2 and NPD constructs in the risk analysis process.

4.2 Risk Assessment and the Non-Persistent Capability Concept

In the next series of diagrams the NPC2 construct is included in the compromise path models and calculations. As with the previous example, this process begins by defining the AG and FOI for the InfoSec Architecture model depicting the Agile Mission group. Inbound connections between the AG and FOI groups are calculated in the models.

In the first arrangement captured in Figure 16, the NPC2 is enforcing the baseline or default separation and sharing profile where connectivity is permitted between Domains AX and AY (as shown by the solid line and position of the conditional connector). There is no direct path between Domains AX and B. To avoid confusion, potential compromise paths are shown with an arrow head. Straightforward

connections are plain lines and included for completeness because they will be used when calculating indirect connectivity.

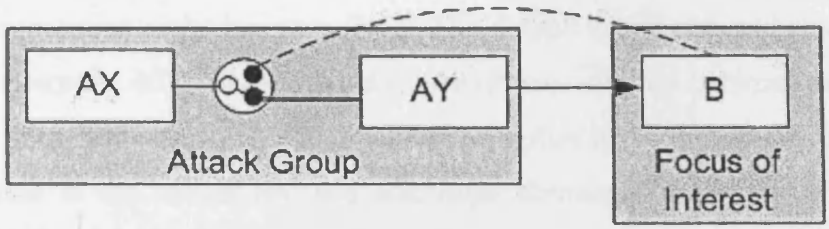


Figure 16: Compromise path model using NPC2 between Domains AX and AY

As shown in Figure 16, when the NPC2 construct is used to switch between Domains AX and AY, only one inbound, direct or single step compromise path exists between the AG and FOI.

Figure 17 shows the same configuration, but in this example the NPC2 construct switches connectivity between Domains AX and B.

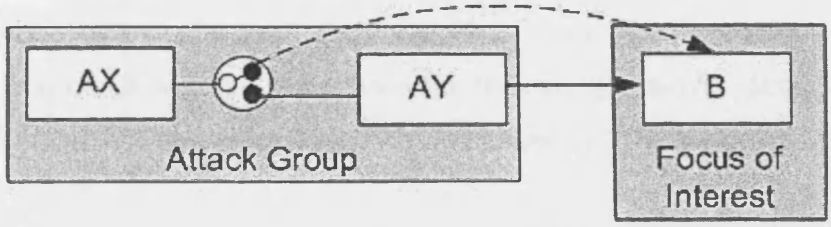


Figure 17: Compromise path model using NPC2 between Domains AX and B

It can be seen from Figure 17 that this configuration yields 2 potential inbound compromise paths between the AG and FOI. The relationship between Domains AY and B has no effect on the relationship between Domains AX and B or AX and AY as controlled by the NPC2 therefore the connections are considered independently.

We have an interest in calculating the rate of growth for potential compromise paths as domains are added to the configuration and believe that the NPC2 can be used to influence both the rate of growth and proliferation of connections. The investigation begins by adding a single domain, Domain C to represent a collaborative partner in the existing model. Recall from the original InfoSec Architecture model in Figure 1,

Section 3.3.1, that the collaborative partner, represented by an external domain, is normally permitted to communicate with Domain B. It is therefore included as a member of the AG and the results, captured in Figure 18 are obtained. The number of potential compromise paths increases by 1. The default separation and sharing policy is in force where the NPC2 is switched to permit connectivity between domains AX and AY (as shown by the solid line). This relationship has no effect on any domains that are added to the model because additional domains are not, in this instance, controlled by the NPC2.

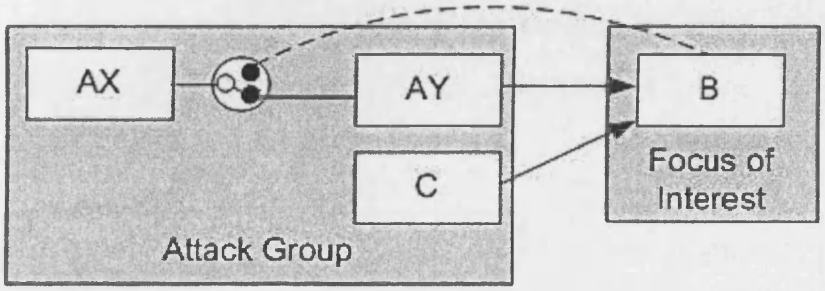


Figure 18: Compromise path model with additional Domain

Finally, Figure 19 shows the case where the NPC2 is switched to permit connectivity between Domains AX and B. Again, we see that the number of direct, single step compromise paths between the AG and FOI increases by 1, when Domain C, is added to the model.

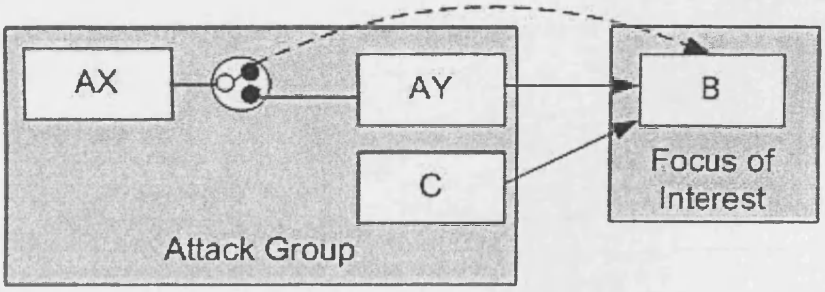


Figure 19: Compromise path model with Domain C

The next series of models deviate quite deliberately from DBSy and consider each domain in the role of potential AG *and* FOI. The DBSy approach defines and bounds the scope of the analysis which enables stakeholders to focus on the part of the system over which they have control. By removing these boundaries and examining the *total*

number of compromise paths arising from *any* permitted connection it is assumed that each domain presents a threat to its immediate and connected neighbour. Clearly this will impact the number of direct compromise paths but we believe that this approach will later prove useful in revealing indirect compromise paths which represent a subtle but very real threat to the system. An analysis of total connectivity will provide additional information which will enhance our understanding of the impact of the NPC2 in reducing the number of potential compromise paths. Calculations for Figure 20 are based on maximum permitted connectivity. This result can be compared with the models in Figures 21 and 22 where the NPC2 is used. In each case, the paths can be totalled quickly and easily by counting the number of arrow heads for each diagram.

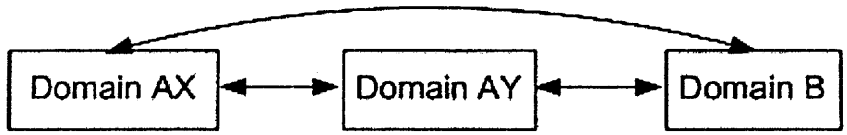


Figure 20: Compromise paths assuming maximum permitted connectivity

Figure 20 shows a total of 6 potential compromise paths when maximum permitted connectivity between domains is assumed.

Figure 21 presents the case where the NPC2 is switched to permit communication between Domains AX and B as shown by the position of the conditional connector. As expected, the number of direct compromise paths is 4.

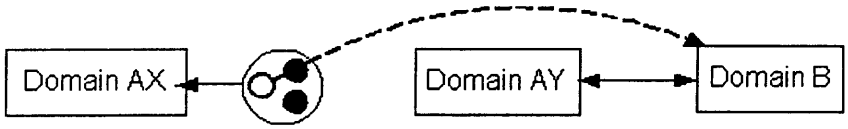


Figure 21: Compromise path model assuming conditional security policy

Figure 22 shows the three-domain arrangement where the baseline separation and sharing policy is in place permitting the connection between Domains AX and AY as depicted by the position of the conditional connector. In this configuration there are a total of 4 direct compromise paths between Domains AX, AY and AY, B

respectively. Note that the temporary connection that *would* exist between Domains AX and B is identified by the broken line if the NPC2 were switched in this direction but is not included in the calculations since it is inactive.

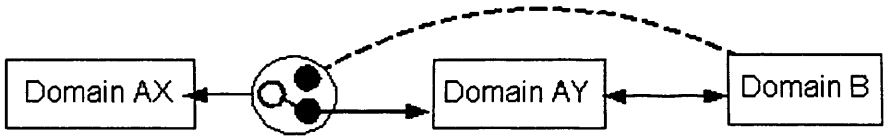


Figure 22: Compromise paths assuming baseline security policy

Figures 23, 24 and 25 all show a configuration of 4 domains where in this case Domain C is permitted full connectivity with existing domains. In Figure 23, maximum permitted connectivity is assumed in the model which provides some results for comparison with a model utilising the NPC2. A total of 12 compromise paths exist.

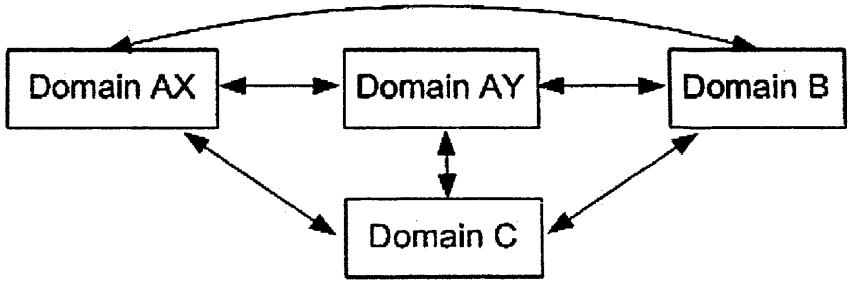


Figure 23: Four domain configuration; maximum permitted connectivity

In Figure 24 the NPC2 restricts connectivity permitting Domain AX the ability to share with Domain B and, in Figure 25 Domain AX is permitted to share information with Domain AY.

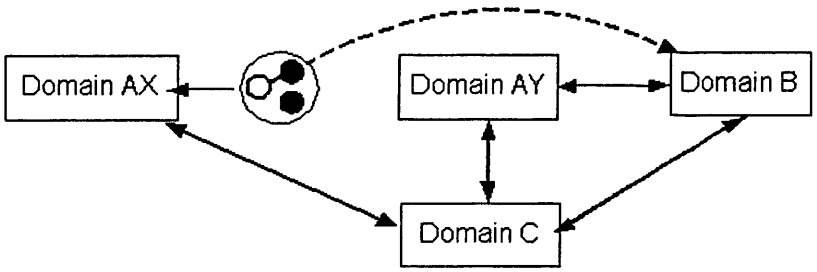


Figure 24: Compromise path model, four domains; NPC2 switches AX and B

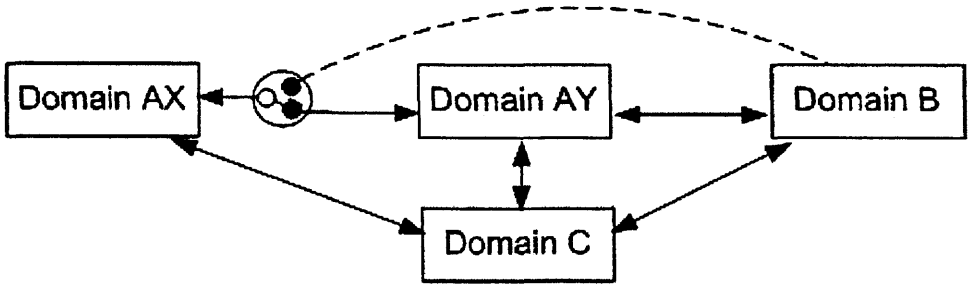


Figure 25: Compromise path model, four domains; NPC2 switches AX and AY

In each scenario there are a total of 10 direct compromise paths.

Figures 26, 27 and 28 are all 4-domain configurations. However, in this series of models Domain C is only permitted partially connectivity with the rest of the system. Figure 26 is the comparison model and incorporates maximum permitted connectivity.

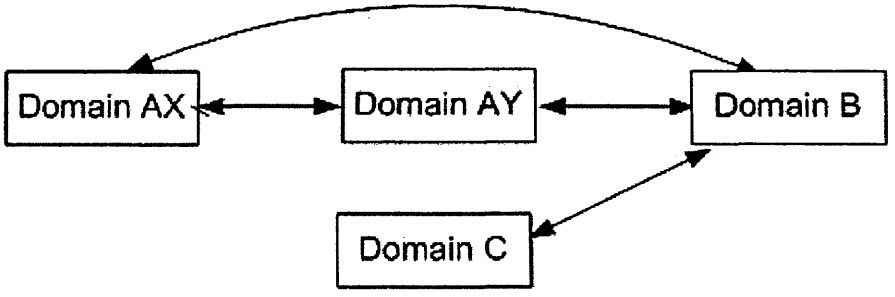


Figure 26: Four domains, partially connected; maximum permitted connectivity

In Figure 27 the NPC2 permits connectivity between Domains AX and B. The result is 6 potential compromise paths.

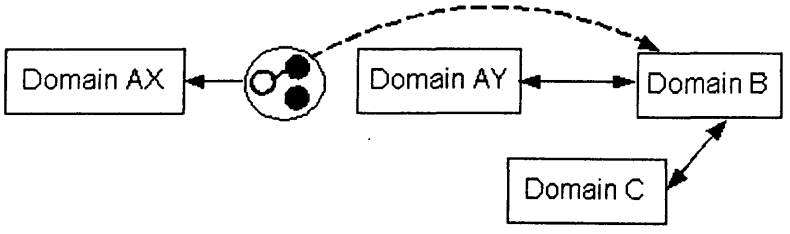


Figure 27: Partial connectivity with fourth domain; NPC2 switches AX and B

Finally in Figure 28 the NPC2 is switched to allow connectivity between Domains AX and AY. Again, there are 6 direct compromise paths.

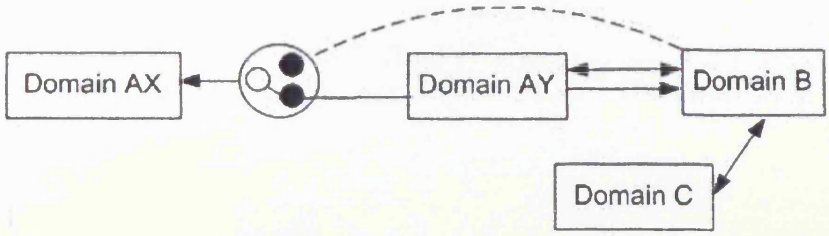


Figure 28: Partial connectivity with fourth domain; NPC2 switches AX and AY

The results from this initial set of models are summarised in Figure 29 which tabulates the potential number of compromise paths in a bar chart.

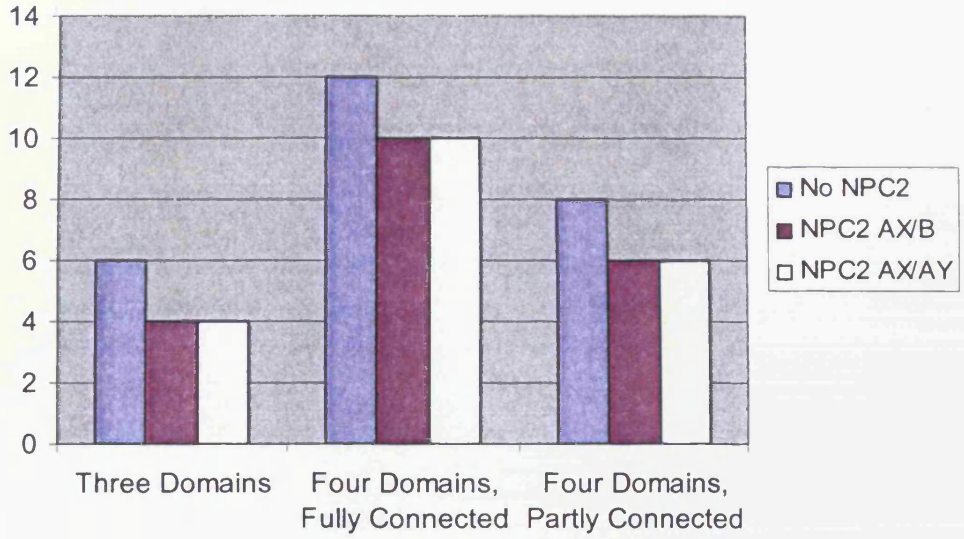


Figure 29: Compromise paths summarised

In Figure 29, the first data set (on the left-hand-side) corresponds to Figures 20, 21 and 22. The second dataset (in the middle) refers to Figures 23, 24 and 25. Finally the third dataset (on the right-hand-side) corresponds to Figures 26, 27 and 28. From Figure 29 it is very clear that for the cases described the NPC2 reduces the number of compromise paths when we consider direct connections. However, this small-scale study also reinforces the importance of looking at *how* domains are connected to one another since this will impact the effectiveness and influence of the NPC2. This is

particularly significant when we consider adding domains to the configuration and calculating the corresponding rate of growth in potential compromise paths.

In our case study, Domain C normally shares data with Domain B only. Therefore we would not expect the rate of growth to be high; an expectation supported by the evidence collected in this section. Full connectivity shows a substantial increase in the number of potential direct paths and we would expect the difference to be higher when indirect connectivity is considered. As shown in Figure 29 the NPC2 will, in some cases, reduce the number of potential direct, inbound compromise paths although it effectiveness is not pronounced in this simple model described. The result is not unexpected since the NPC2 provides an 'exclusive OR' between two domains. The effect of the NPC2 when domains are added to the model is dependent on the relationship between the NPC2 and those additional domains. If, for example, Domain C had also been controlled by the NPC2 acting either in the role of a multi-way connector or an additional construct the results would probably have been different. The use of a multi-way NPC2 is not explored in this thesis. However, it could be an interesting development in our research and should be investigated as future work. Multiple 2-way NPC2 constructs are studied in Chapter 7.

4.3 Risk Assessment and the Non-Persistent Domain

In this section we calculate the effect on potential compromise paths when the NPC2 and NPD constructs are used together. Recall, that the NPD is designed to capture the temporary formation of a domain. It is only formed under specific conditions otherwise a regular domain would be used. It is therefore suited to ad-hoc networks. For convenience Figure 10, the revised version of the mission group including our suggested notation for the NPD, is repeated in Figure 30.

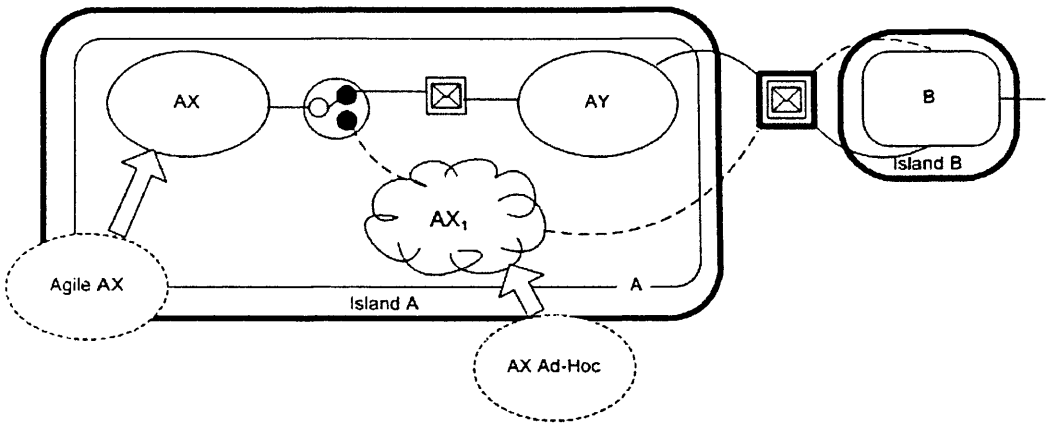


Figure 30: Scenario including NPD

The InfoSec Architecture model is redrawn to show inbound, direct compromise paths between the AG containing Domains AX, AX_1 and AY and the FOI containing Domain B. Maximum permitted connectivity is assumed. The potential compromise path arising from the connection between Domains AX_1 and B is included.

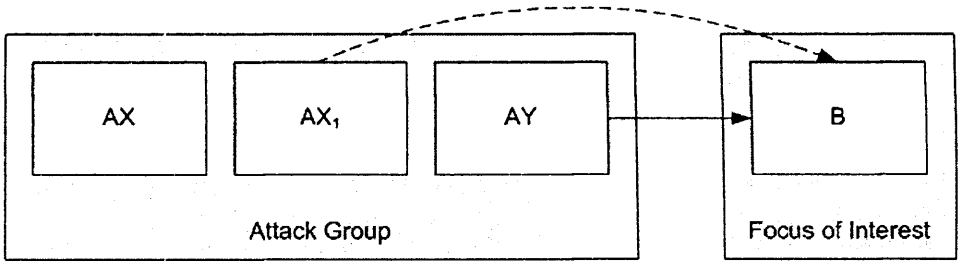


Figure 31: Compromise path model assuming maximum permitted connectivity

Figure 31 shows a total of 2 direct compromise paths between the AG and FOI. The model makes no account of potential paths opened through indirect connections which we know, in this case, exist between Domains AX to AY and when it is active, from Domain AX to AX_1. We conjecture that when indirect paths are included the number will increase significantly. Another factor affecting this model is the inclusion of the path between AX_1 and Domain B, which again only exists if this temporary domain is active. The model is adjusted to show the NPC2 controlling connectivity between domains AX, AX_1 and AY and the paths redrawn accordingly. This configuration is captured in Figure 32.

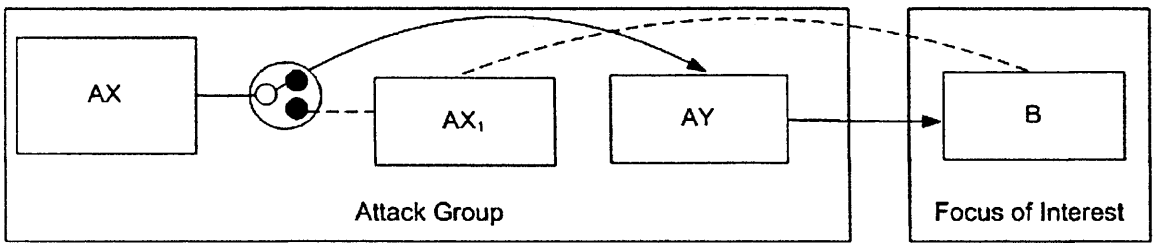


Figure 32: Compromise path model controlled by NPC2

Figure 32 indicates that only 1 direct inbound compromise path exists between Domains AY and B. When the NPC2 is switched to allow connectivity between AX and AX_1 the total number of direct compromise paths will be 2. This is the same as the maximum permitted connectivity model in Figure 30 since the path between Domains AX_1 and B will be activated.

The NPD has an additional benefit to our case study and that is managing risk internal to the domain itself, which has not been specifically addressed so far. This internal risk arises because the case study assumes that domain members do not have *equal* capabilities all of the time. This assumption is fundamental to our research because it considers temporal and event driven changes to separation and sharing capabilities. DBSy assumes that all members of a domain are able to share without restriction. It is only *external* connections that may be subjected to control. Internal risk to the domain would therefore not normally be an issue. In our approach the scenario has been complicated because a single member of Domain AX has the temporary elevation of capabilities allowing them to share with Domain B. Given that this feature is conditional, the subsequent change to the separation and sharing profile may, in reality, never occur and this member will simply be a regular occupant of Domain AX. The change to potential compromise paths has been considered for the case where the NPC2 permits connectivity between Domains AX and B. However, neither the affect of this change on the remainder of AX, nor the affect if the NPC2 is switched back to the default separation and sharing policy, has been studied. The NPD could be used to identify this exceptional user since under this regime, they would move from a regular domain into a temporary structure. Figure 30 shows that Domain AX and the NPD AX_1 each have their own Environment and Portal for domain access, the two entities are therefore quite separate.

If Domain AY were to become available again, the default separation and sharing policy permitting Domain AX to share with Domain AY but not Domain B would be enforced. The temporary elevation in capability would be revoked from our NPD member and Domain AX_1 would effectively cease to exist. Revocation of capability is of course a fundamental issue in this respect. The implementation of our solution must ensure that the removal of authorisation is clean. It is expected that this would be tested through auditing resource access both before and after the change in policy. This could be viewed in the audit or log files. Depending on the features of the security infrastructure it may also be possible to view access control lists against the resource in question. If information has been exchanged between users there must be a means of securing it from the user once their privileges have been revoked in order to prevent unauthorised, intentional or accidental disclosure or modification to the data. This is not a trivial task since the user could have already saved information to a location where they have privileges. Of course, the human in the loop presents a potential risk in any security solution. They may simply remember information and not record it anywhere in which case no technical measure will be of use. The threat presented by the user will however be taken into consideration during the threat and risk analysis process with knowledge of the level trust which may be vested in them.

So far the existence of a compromise path arising from internal risk has not been identified in either the models or incorporated into the calculations. This raises an important question; at what stage should it be included in this process? The issue will only occur if domain members have (temporarily) different capabilities from one another. When the NPC2 is either switched back to the default separation and sharing policy or indeed moves on to a completely new and as yet unknown configuration it may not exist at all. Is it therefore appropriate to include it at all? On closer inspection, it is concluded that this issue is not the prerogative of the NPC2. Since users may be members of more than one domain, by switching between them, there is always the opportunity for introducing compromise. This is one reason of course why DBSy shows separation between the domains and the means of access. We therefore conclude that judicious use of the NPD overcomes some of the concern of introducing internal risk to a domain since the user is temporarily moved into a new domain. A new method of access is provided thus separating them from the domain of origin. We therefore introduce no greater risk through the non-persistent constructs than the

accepted method of allowing a user to work in more than one domain. In fact it could be argued that less of a risk is introduced with this approach since the user does not have the capability of the temporary domain unless the conditions trigger the change in security policy which specifically allows it.

4.3.1 Variation to model using Temporary Business Connector only

Our case study shows a dependency between Domains AX, AY and B whereby simultaneous connectivity is not permitted between AX and the other domains. The conditional connector is used as an 'exclusive OR' to switch between domains in accordance with the separation and sharing policy. For our notion of fit for purpose security, changes to the policy are required because of a condition in the environment and may not affect the relationship between existing domains at all. Consider the case where the default security policy permits Domain AX to share with Domain AY. Under certain circumstances the permission is extended to Domain B, but this is only on a temporary basis and has no dependency on Domain AX's relationship with Domain AY. Figure 33 shows the InfoSec Architecture model for this arrangement.

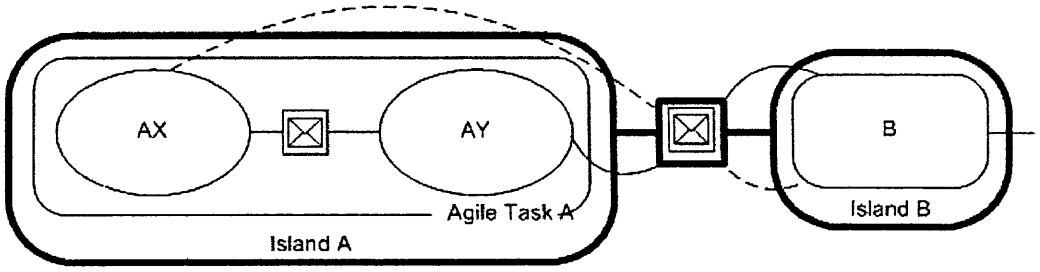


Figure 33: Conditional connectivity omitting dependencies between domains

Recall from Chapter 3 that the NPC2 is actually comprised of the conditional connector represented as a 'switch' and temporary business connector combined. The temporary business connector may be used on its own where connectivity between domains is not mutually exclusive. This is exactly the case shown in Figure 33 where we omit the conditional connector from the model because the exception to the normal separation and sharing profile is not dependent on the relationship between Domains AX and AY. In terms of risk analysis this variant would need to be considered as normal and exceptional states where the exceptional state could also be

considered as maximum permitted connectivity. Figures 34 and 35 model inbound compromise paths for the case where the agile mission group containing Domains AX and AY is the potential AG and Domain B is the FOI.

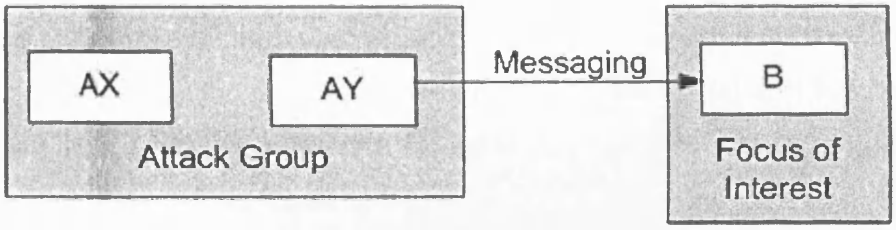


Figure 34: Compromise path model baseline security policy, AG and FOI defined

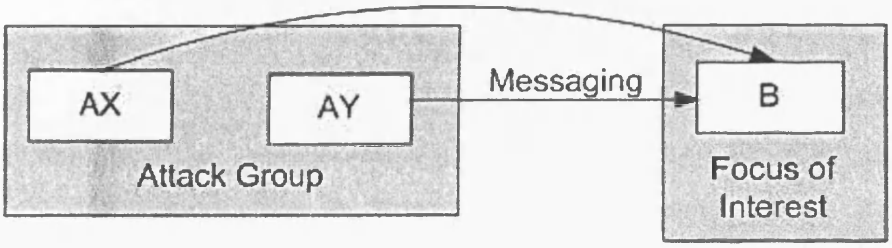


Figure 35: Compromise path model exception to policy, AG and FOI defined

As with our earlier models, the connection between Domains AX and AY is not included although, for the reasons described previously, it is proposed that some reference should be made to this relationship since it does form a potential indirect compromise path. In Figure 34 there is only 1 inbound direct compromise path because the connection between Domains AX and B is not permitted at this time. Figure 35 yields the same results as our first in example back in Figure 15, because it makes no concession for the temporary nature of the connection between Domains AX and B. A total of 2 potential direct compromise paths exist for this model

Finally, for the case where all domains are viewed in the capacity of both AG and FOI the results in Figures 36 and 37 are obtained.

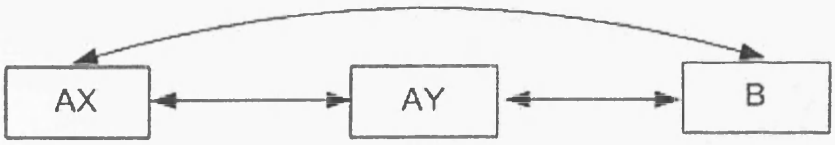


Figure 36: Compromise path model conditional security policy

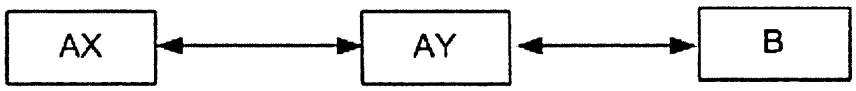


Figure 37: Compromise path model baseline security policy

Figure 36 shows a total of 6 potential direct compromise paths because maximum permitted connectivity is assumed; whereas Figure 37 has a total of 4 because in this case the business connection between Domains AX and B is inactive. Interestingly these results are exactly the same as using the NPC2 and placing some condition on permitted connectivity through Domain dependencies although of course the configuration and conditions for this variation is different.

The results support our assertion that differentiating between the default and exceptional permitted connections provides a more accurate view of the number of potential compromise paths. We conclude that this information will ultimately affect the way in which the security engineer mitigates risk through system controls.

So far, the work in this chapter has focused on direct compromise paths and, crucially, connectivity introduces indirect or covert paths, which must also be analysed and addressed. In the next section all of these initial results are used to form the basis of an investigation which looks specifically at higher order connectivity. This allows us to understand the potential impact of the NPC2 and NPD constructs in greater depth.

4.4 Calculating Indirect Compromise Paths

Indirect compromise paths are a natural consequence from permitted business connections between domains and neither their presence nor impact may be immediately obvious. It is therefore important to have an efficient method of identifying indirect compromise paths so that their risk can be understood and mitigated. It has already been shown that the number of direct compromise paths grows as domains are added to the network, although the rate of growth is dependent on the richness of connections. However, this will also impact the number of indirect connections between domains. It is our opinion that the affect of indirect connectivity can be substantially reduced through judicious use of the NPC2 or NPD to reflect the temporary and conditional nature of connections and domains where applicable. To

assist with the analysis, some ideas from network theory are applied to the various domain configurations.

According to Chartrand, a network can be defined as "... a graph or directed graph (digraph) together with a function which maps the edge set into the set of real numbers ..." [20, p.19]. In this chapter it is sufficient to employ either a graph or directed graph to complement the DBSy models.

A part of network theory is concerned with connectivity and the identification of potential paths through any type of network. In the disciplines of computing and communications applications of the approach could include, amongst others, the calculation of routing efficiency using shortest path algorithms, designing redundant paths for quality of service and the determination of node accessibility. As stated previously our work uses the notion of connectivity and accessibility as a measure of vulnerability between domains. Using graphs to calculate node accessibility is therefore of particular interest. Although connectivity is normally considered at the individual node level, for the purposes of this investigation it is sufficient to continue working at a higher level. The Domain is seen as a collection of nodes with a common security profile and function and can therefore be regarded as a single entity.

Graphs provide a convenient method of representing the DBSy InfoSec Architecture models. Using graph notation *vertices* or *nodes* are drawn as small circles and used to represent the DBSy domains. This thesis uses the term *node(s)* in preference to *vertex* or *vertices*. An *edge* is shown as a line joining two nodes and indicates the DBSy permitted business connections. Domain accessibility is calculated by the *degree* of the node. In graph theory, a *degree* is the number of edges of a graph incident upon the node [20, p.27] or for our purposes, the number of permitted business connections for a domain.

Matrices can be used to represent a graph. Connectivity or potential compromise paths can therefore be captured within the matrix. This can be used subsequently to calculate higher-order connectivity between nodes using matrix multiplication. Matrices provide a convenient method to present the connectivity status and means of assessing the impact of multi-hop or indirect connections.

In graph theory, the path length is defined as the number of nodes that are traversed between the starting and final node in a path. Therefore, a direct or single path connection between two nodes is the equivalent of a path length of 1 because there are no intervening nodes. Indirect or multi-hop connections indicate a path length greater than 1. In the example below, Figure 38 shows the DBSy model of our agile mission group redrawn as a graph. The individual connections between AX with AY and AX with B both have a path length of 1. The path between AX and B (via Domain AY) gives a path length of 2.

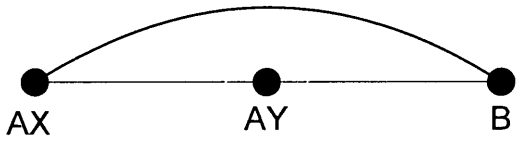


Figure 38: Basic scenario represented as Graph

In this research the matrix is defined as follows. For a set of domains D_i i=1,2,.....,N the ij-th entry in the $N \times N$ connection matrix C is

$$\mathbf{C}_{ij} = \begin{cases} 1 & \text{if } D_i \text{ is connected to } D_j \\ 0 & \text{if } D_i \text{ is not connected to } D_j \\ 0 & \text{if } i = j \end{cases}$$

The graph shown in Figure 38 can be represented by the connectivity matrix C_1 in (2) for domains $\mathbf{D}_1$ =[AX, AY, B].

$$C_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad \ldots (2)$$

This model shows maximum permitted connectivity, that is, all of the edges between nodes are included irrespective of their state. In (3) connectivity for the case C_1^2 is

shown where the path length or hop-count is 2. It can be seen that the number of potential paths increases to 12; double that of C_1.

$$C_1^2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \quad \ldots (3)$$

Total connectivity up to and including second order, C_{1tot} is calculated in (4) and gives a total of 18 potential compromise paths.

$$C_{1tot} = C_1 + C_1^2 = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \quad \ldots (4)$$

(4) shows quite clearly that in this configuration, each domain is equally accessible.

The next case in Figure 39 considers the addition of Domain C to the model. Like the first example in Figure 26, Domain C is partially connected and may only share with Domain B. The Domains in Figure 39 are $\mathbf{D}_2$= [AX, AY, B, C].

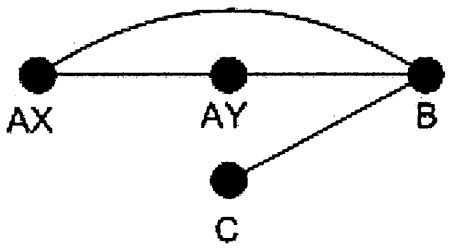


Figure 39: Graph showing partial connectivity including Domain C

This configuration is for maximum permitted connectivity and does not distinguish between a permanent and temporary connection involving Domains AX, AY and B. The connectivity matrix C_2 is shown in (5).

$$C_2 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \ldots (5)$$

C_2 confirms our original findings that when there is no differentiation between non-persistent and permanent connections, the number of potential compromise paths is 8. Domain B is the most accessible or connected node.

(6) shows connectivity for the case C_2^2 where the path length is equal to 2.

$$C_2^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 3 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \ldots (6)$$

In (6) it can be seen that the number of potential compromise paths has risen from 8 to 18. The potential richness of connectivity between the nodes is much greater once indirect links are considered. Note, for example, that according to (5) Domain B does not have connectivity to itself. Once indirect connections for a 2-hop configuration are calculated, there are three potential routes via AX, AY and C respectively.

Total connectivity for this configuration is given in (7) and yields 25 potential paths.

$$C_{2tot} = C_2 + C_2^2 = \begin{bmatrix} 2 & 2 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 2 & 2 & 3 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad \ldots (7)$$

(7) shows that Domain AY is the most accessible or vulnerable domain in this case. If the path length was increased to 3 hops, that is, the case C_2^3, we postulate that the number of connections would again increase.

The final example in this set of models, is the four domain fully connected network. This is shown as a graph in Figure 40 for domains $\mathbf{D}_3$ = [AX, AY, B, C]. The resulting connectivity matrix C_3 is given in (8).

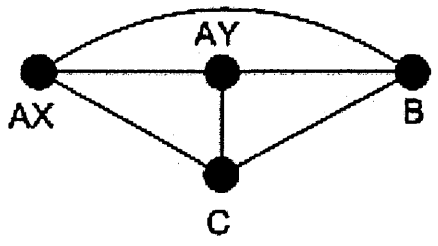


Figure 40: Four domain, fully connected configuration represented as Graph

$$C_3 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad \ldots (8)$$

C_3 yields a total of 12 direct potential compromise paths. Indirect connectivity is shown in (9) for the connectivity matrix C_3^2.

$$C_3^2 = \begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix} \quad \ldots (9)$$

When the path length is equal to 2, the number of potential compromise paths increases from 12 in (8) to 36 in (9). The total number of potential compromise paths is calculated in (9) giving 48 connections.

$$C_{3_{tot}} = C_3 + C_3^2 = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{bmatrix} \quad \ldots (10)$$

The rate of growth for direct and indirect compromise paths, where maximum permitted connectivity is assumed, is summarised in the graph in Figure 41.

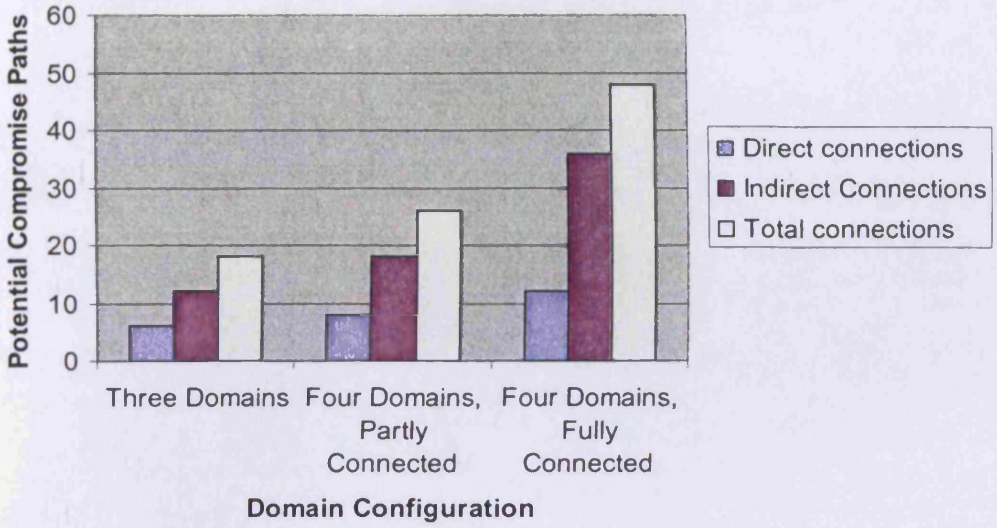


Figure 41: Comparison of direct and indirect connectivity

Figure 41 shows a sharp increase in the number of connections between direct and indirect connections even though the calculations have only considered a path length of 2. The analysis could continue to look at higher order connections using the same process of matrix multiplication. It is expected that the increase and richness of connections would become even more pronounced.

A fundamental part of this study involves analysing and articulating the impact of the NPC2 construct within the scope of the connectivity matrix and assessing whether it reduces the potential number of compromise paths when used. In Section 3 the equivalent connectivity diagrams were shown but used the NPC2 construct to switch between different domains. When the model is shown as a graph there is no means of identifying the temporary or conditional nature of a connection. The connection can only be included in the graph or omitted. To avoid confusion, the compromise path model and connectivity state is shown alongside the relevant graph. In all cases all connections are considered as potential compromise paths, therefore neither an AG nor FOI are specified.

The first model shown in Figure 42 is the three domain configuration where the security policy is in its baseline state. The conditional connector permits connectivity between Domains AX and AY.

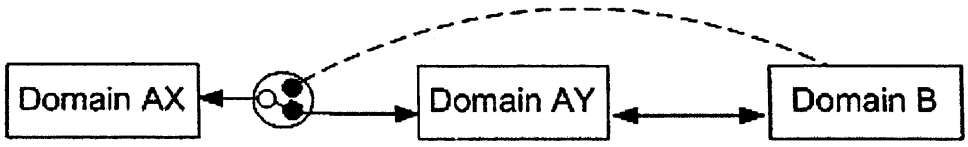


Figure 42: NPC2 switching Domains AX and AY

This is redrawn as a graph in Figure 43, but no edge is shown between AX and B because this path is not currently active.

Figure 43: Configuration in Figure 42 redrawn as a Graph

In Section 4.2 the number of potential compromise paths for this model was calculated as 4. The connectivity matrix C_4 in (11) confirms the original findings.

$$C_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \ldots (11)$$

Indirect connectivity for a path length of 2 is shown for the case C_4^2 in (12) where there are a total of 6 potential compromise paths.

$$C_4^2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad \ldots (12)$$

Total connectivity for this model is given in (13) which yields 10 potential compromise paths.

$$C_{4tot} = C_4 + C_4^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \ldots (13)$$

Figure 44 shows the exceptional case where NPC2 permits a business connection between Domains AX and B only. Connectivity for the equivalent graph shown in Figure 45 is given in (14) where there are a total of 4 potential compromise paths.

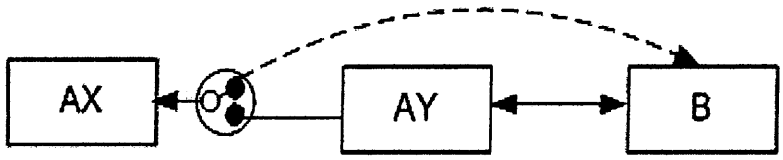


Figure 44: Compromise path model; NPC2 permits sharing between AX and B

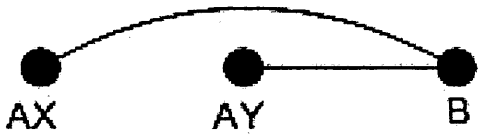


Figure 45: Graph for model in Figure 44

$$C_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \ldots (14)$$

Indirect connectivity for the case C_5^2 is given in (15) totalling 6 potential paths.

$$C_5^2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \ldots (15)$$

The total for this configuration is calculated using (16) and shows a total of 10 potential compromise paths.

$$C_{5tot} = C_5 + C_5^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \end{bmatrix} \quad \ldots (16)$$

This result is the same as (13). In both models the NPC2 has reduced the number of paths when compared to a model where maximum permitted connectivity is calculated.

In the next series of diagrams the case where an additional Domain has been added to the configuration is presented. The domains are $\mathbf{D}_5$ = [AX, AY, B, C]. It is known from (4) that Domain C is not controlled by the NPC2 therefore little change is observed in the number of direct potential compromise paths when this domain is included. However, when higher order connectivity is calculated it is expected that its presence will have a greater impact. In Figures 46 and 47 the NPC2 permits connectivity between Domains AX and AY, which prohibits simultaneous connectivity between Domains AX and B.

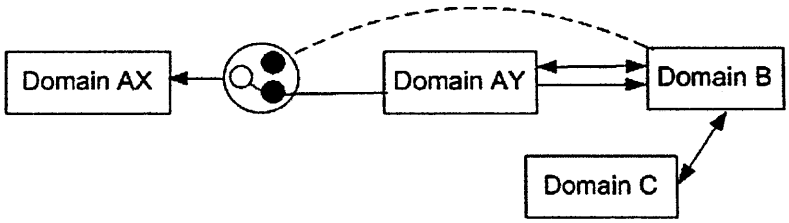


Figure 46: Four domain configuration utilising NPC2

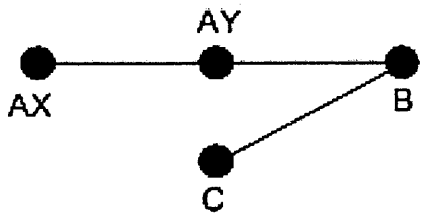


Figure 47: Four domain configuration in Figure 46 redrawn as Graph

The connectivity matrix for direct and indirect potential compromise paths are shown in (17) for C_5 and (18) for C_5^2 respectively.

$$C_5 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \ldots (17)$$

$$C_5^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \ldots (18)$$

$$C_{5tot} = C_5 + C_5^2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \ldots (19)$$

Connectivity for C_5 shown in (17), gives a total of 6 potential compromise paths. This increases to 10 for C_5^2 when the path length is increased to 2, as shown in (18). The matrices reveal that Domains AY and B are the most connected and therefore, seemingly, the most vulnerable. The contents of C_5 and C_5^2 are summed in (19) to obtain the *total* connectivity for this configuration. This yields 16 potential compromise paths.

(20) shows the connectivity matrix C_6 for the configuration in Figures 48 and 49. Here the NPC2 permits connectivity between AX and B which denies connectivity between AX and AY.

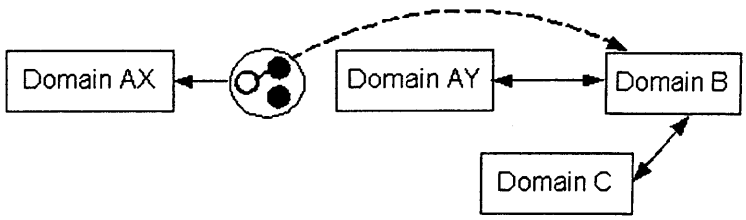


Figure 48: Four domain partial connectivity using NPC2

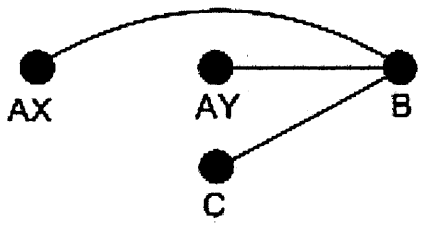


Figure 49: Figure 48 represented as Graph

$$C_6 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \ldots (20)$$

In (20) C_6 shows of total of 6 potential compromise paths where Domain B has the greatest connectivity. This increases to 12 for the case C_6^2 as shown in (21).

$$C_6^2 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 3 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \ldots (21)$$

Total connectivity is given in (22) yielding 18 potential compromise paths.

$$C_{6tot} = C_6 + C_6^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \ldots (22)$$

Figure 50 summarises the results of this section in a bar chart. It refers to the *total* number of potential compromise paths for each configuration, that is, the summation of direct and indirect connections for each case.

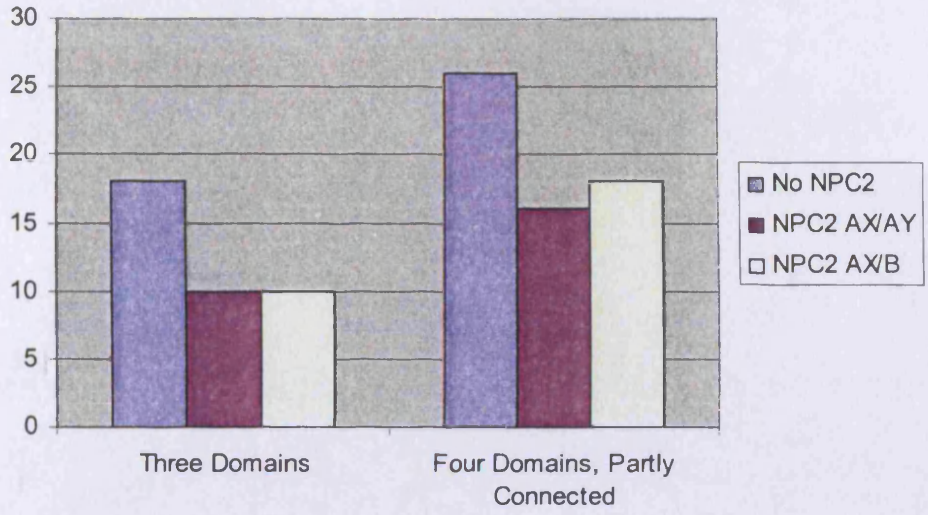


Figure 50: Summary of indirect connectivity with and without the NPC2

In Figure 50 the first data set on the left-hand side refers to (4), (13) and (16). The second data set on the right-hand side refers to (7), (19) and (22).

Figure 50 shows quite clearly that in each case the NPC2 construct has a significant impact in reducing the number of potential compromise paths. Furthermore, it is hypothesised that its usefulness will become more marked as higher orders of connectivity are considered and/or the numbers of domain increase. The degree of impact is affected by the placement of the NPC2 and of course the level of permitted connectivity between additional domains in the model.

To complete this section indirect connectivity and the NPD are considered. Recall from Section 4.3 that when direct connectivity was calculated use of the NPD presented little difference in the potential number of compromise paths irrespective of when the NPC2 was used. However, we postulate that this position will change as higher order connectivity is calculated. The figures and matrices that follow contain the results of our study. Like the preceding models there is no differentiation between inbound and outbound connections involving the AG and FOI; any level of connectivity is seen as a potential compromise path.

In order to determine the effect of introducing the NPD, our initial risk calculations are based on maximum permitted connectivity. Figure 51 captures this configuration

for domains $\mathbf{D}_7$ = [AX, AY, AX_1, B]. Note that the temporary or conditional connections are defined using a broken line to indicate an exception to the baseline policy. The connectivity matrix C_7 is given in (23) and shows 8 potential direct compromise paths.

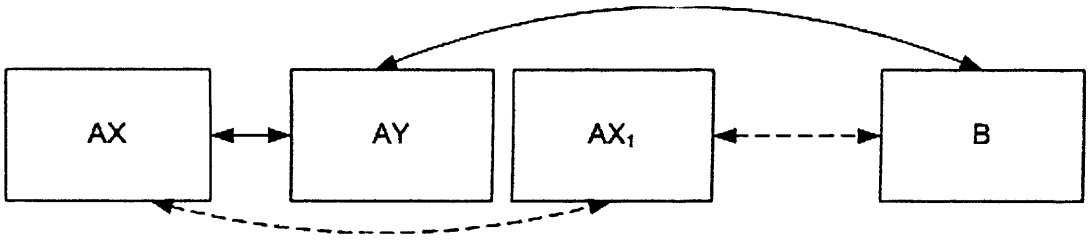


Figure 51: Compromise paths and the NPD, maximum permitted connectivity

$$C_7 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \ldots (23)$$

In (24) this number increases to 16 for C_7^2.

$$C_7^2 = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix} \quad \ldots (24)$$

Total connectivity is calculated in (25) and shows 24 potential compromise paths.

$$C_{7tot} = C_7 + C_7^2 = \begin{bmatrix} 2 & 1 & 1 & 2 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 2 \end{bmatrix} \quad \ldots (25)$$

The configurations are recalculated using the NPC2 and NPD constructs to determine the *actual* number of potential compromise paths for the case where the security policy does not allow participation with the NPD. C_8 is the connectivity matrix for the

default policy where Domain AX is permitted to communicate with Domain AY. A total of 4 direct potential compromise paths are shown in (26), which is a 50% reduction in the number of paths when (26) is compared with (23).

$$C_8 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \ldots (26)$$

For the case C_8^2, (27) shows an increase of only 2 connections, from 4 to 6. This gives an overall total of 10 potential compromise paths when (28) is computed.

$$C_8^2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \ldots (27)$$

$$C_{8tot} = C_8 + C_8^2 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \ldots (28)$$

In the case where Domain AX is not permitted to share with Domain AY the NPD AX_1 is established. The connectivity matrix C_9 is shown in (29).

$$C_9 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \ldots (29)$$

(29) shows a total of 6 direct compromise paths. Although this is clearly less than the worst case scenario depicted in (23) where maximum permitted connectivity was assumed, it is also greater than (26) where the security policy is in its default state. This is not unexpected since an additional domain is being created and connected,

albeit temporarily, to accommodate the 2IC in this case study and maintain connectivity with Domain B.

(30) is the result of calculating C_9^2 and gives 10 potential compromise paths.

$$C_9^2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{bmatrix} \quad \ldots (30)$$

Total connectivity for this configuration is calculated in (31) and gives 16 potential compromise paths

$$C_{9tot} = C_9 + C_9^2 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \quad \ldots (31)$$

The total number of potential compromise paths as computed in (25), (28) and (31) are summarised in a bar chart in Figure 52.

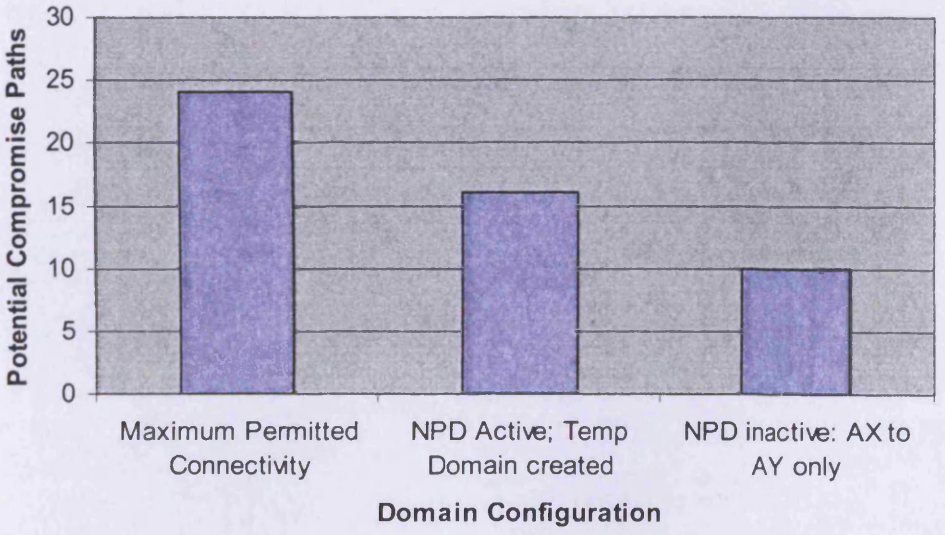


Figure 52: Summary of compromise paths for configurations using the NPD

Figure 52 shows that the use of the NPC2 and NPD constructs may result in a significant reduction in the potential number of compromise paths when compared with calculating maximum permitted connectivity. We have always maintained that the result is not unexpected because the NPC2 and NPD enable us to accommodate change whilst recognising that the condition requiring that change may not occur. It is therefore, in our opinion, a more realistic representation of the *actual* number of compromise paths for states we are able to anticipate.

4.5 Combining Connectivity States within a Single Matrix

In Chapter 3 the difficulty of capturing temporary, changing or conditional connections in the graphical models was observed. Our solution, which lies in the NPC2 and NPD notation, allowed us to combine multiple states onto a *single* InfoSec model. This benefit has not, until now, been realised in the connectivity matrices.

Our analysis currently uses two separate matrices to represent each state of the NPC2 using a numeric '1' or '0' to symbolise the connection states 'connected' or 'not connected' respectively. This information is slightly misleading in that it cannot capture the *potential* for connectivity. Using a '0' could suggest that a path is *never* connected which in our example, is not the case. It is possible to imply potential connectivity in a single matrix if we diverge from the use of a '1' or '0' and apply other symbology. Given that the NPC2 has been used as a Boolean true or false, it follows that Boolean algebra could also be incorporated into the matrices to capture the state of the connection. Although this option has not yet been fully explored, preliminary findings, as described in this section, look promising.

The InfoSec Architecture model in Figure 53 is a repeat of Figure 7 and shows that the NPC2 permits sharing between Domains AX and AY.

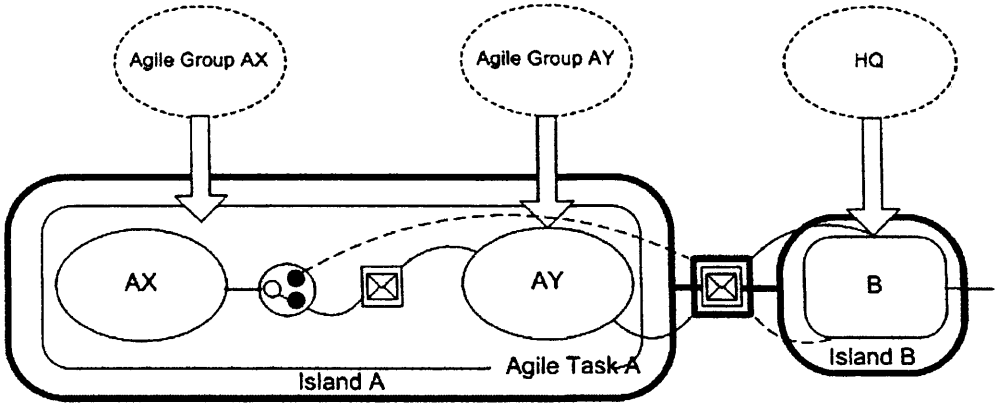


Figure 53: Basic InfoSec Architecture model

From previous calculations it is known that the connectivity matrices C_4 and C_5 are obtained from this model when the NPC2 permits sharing between Domains AX and AY or AX and B respectively. The configuration in Figure 53 is reused to demonstrate that the two conditional connection states can be combined into a single matrix using a combination of real numbers and a Boolean representation of the NPC2. The accuracy of the results can be confirmed against C_4 and C_5. Although our approach is unusual it should enable us to clearly differentiate between potential and conditional (active or inactive) and regular business connections.

Combining Boolean and regular values requires two different types of addition when summing the contents of the matrix. Numerical addition is used for regular numbers. Boolean conjunction or disjunction is applied to the Boolean values to obtain a regular number that can be included in the numerical addition. In all of the matrices involving Boolean values and regular numbers the following conditions are observed. The conditional connection can take the Boolean value of b or $\overline{b}$ as follows:

- b represents the *conditional* (active) path between two domains as controlled by the NPC2. For the purposes of calculating connectivity, $b = 1$.
- $\overline{b}$ represents the *conditional* (inactive) path between two domains as controlled by the NPC2. For the purposes of calculating connectivity, $\overline{b} = 0$.

When $b = 1$, $\overline{b} = 0$ and vice versa.

The Boolean values are used with the regular numbers '1' or '0' where:

- 1 represents an active (permitted) connection
- 0 means that no connection is permitted

The matrix C_{10} shown in (32) is obtained when the Boolean values and ordinal numbers describe the connectivity state in Figure 53 for domains $\mathbf{D}_{10}$ = [AX, AY, B]. In this example Domain AX has a conditional (active) business connection with Domain AY and a conditional (inactive) connection with Domain B.

$$C_{10} = \begin{bmatrix} 0 & b & \bar{b} \\ b & 0 & 1 \\ \bar{b} & 1 & 0 \end{bmatrix} \quad \ldots (32)$$

Total connectivity is achieved by summing the contents of the matrix. Equation (33) shows the sequence of calculations for obtaining the total number of potential compromise paths for this configuration.

$$\begin{aligned} total &= 2b + 2 + 2\bar{b} \\ total &= 2(1) + 2 + 2(0) \\ total &= 2 + 2 = 4 \end{aligned} \quad \ldots (33)$$

The first line in (33) shows the total number of regular and conditional compromise paths. The second line simplifies the calculation by substituting the values of $b = 1$ and $\bar{b} = 0$ into the equation. The third line performs the addition of 4 potential compromise paths. This total corroborates the results obtained in C_4.

It is known that the conditional connection may only be b or $\bar{b}$. Furthermore, when $b = 1, \bar{b} = 0$ and vice versa. Total connectivity for the alternate configuration, that is, when Domain AX is permitted to communicate with Domain B, may be calculated by one of two methods. A separate matrix can be constructed where the existing Boolean value is replaced with its complement or a single matrix can be reworked by substituting the alternative values for b and $\bar{b}$.

We illustrate the second method in (34) where $b = 0$ and $\overline{b} = 1$.

$$\begin{aligned} total &= 2b + 2 + 2\overline{b} \\ total &= 2(0) + 2 + 2(1) \\ total &= 2 + 2 = 4 \end{aligned} \qquad \ldots (34)$$

Our initial conclusion is that the use of a symbol as opposed to a numeric is more representative of the actual state of connectivity. By using b to represent an active (conditional) connection with the NPC2 we know that the path is permitted. Where $\overline{b}$ is used the connection *may be* permitted but is not *currently* permitted. The '0' and '1' are literal states and therefore provide no information regarding the temporary or changing nature of the connection. Whilst it is usually necessary to calculate the potential number of compromise paths individually for each connection, there is a short cut. In the expression $total = 2b + 2 + 2\overline{b}$, it is possible to group b and $\overline{b}$ together and use the Boolean identity $b + \overline{b} = 1$ to obtain $total = 2(b + \overline{b}) + 2 = 2 + 2 = 4$ *whatever the value of* b *and* $\overline{b}$, in agreement with (34) and (35). This underlines the value of persisting with the algebraic representation rather than pursuing two separate evaluations.

The earlier models using '1' or '0' enable us to calculate indirect connectivity using matrix multiplication. This is still possible in the combined matrix and means that the level of complexity and detail of information held within the matrices can be enhanced whilst retaining a uniform methodology to calculating higher order connectivity. The connectivity shown in Matrix C_{10}^2 in (35) is obtained when C_{10} is multiplied by itself and represents a path length of 2 between domains.

$$C_{10}^2 = \begin{bmatrix} 0 + b^2 + \overline{b}^2 & \overline{b} & b \\ \overline{b} & b^2 + 1 & b\overline{b} \\ b & \overline{b}b & \overline{b}^2 + 1 \end{bmatrix} \qquad \ldots (35)$$

In (35) and all other matrices where both Boolean variables are used and higher order connectivity is calculated, the substitutions $b^2 = b, \overline{b}^2 = \overline{b}$ are made. This allows

simplification of the matrices and addition of their contents. For ease of understanding all stages of the simplification process have been shown for our first example in matrix C_{10}^2 in (36) and (37). Subsequent examples will not show all of the stages but the final version of the matrix after simplification.

The matrix in (36) shows connectivity after C_{10}^2 is simplified by substituting for b^2 and $\overline{b}^2$.

$$C_{10}^2 = \begin{bmatrix} 0+b+\overline{b} & \overline{b} & b \\ \overline{b} & b+1 & b\overline{b} \\ b & \overline{b}b & \overline{b}+1 \end{bmatrix} \quad \ldots (36)$$

In (37) the content of C_{10}^2 has been further simplified by substituting $b+\overline{b}=1$ and $b\overline{b}=0$.

$$C_{10}^2 = \begin{bmatrix} 1 & \overline{b} & b \\ \overline{b} & b+1 & 0 \\ b & 0 & \overline{b}+1 \end{bmatrix} \quad \ldots (37)$$

Total connectivity of the matrix is calculated in (38) where 6 potential compromise paths are achieved when $b=1$ and $\overline{b}=0$. This result corroborates C_4^2. As with (33) all stages of the addition process have been included.

$$\begin{aligned} total &= 2b+2\overline{b}+(b+1)+(\overline{b}+1)+1 \\ total &= 3b+3\overline{b}+3=6 \end{aligned} \quad \ldots (38)$$

This is a simple example with only two states. In Figure 54 a variation of the model is used which includes an additional NPC2 between Domains B and D. The business model has been simplified to remove the method of sharing from the connections. There is no relationship between the NPC2 constructs, that is, connectivity between Domains A and B is not dependent on the state of connectivity between Domains B and D. Each NPC2 must be identified and calculated independently in the matrix.

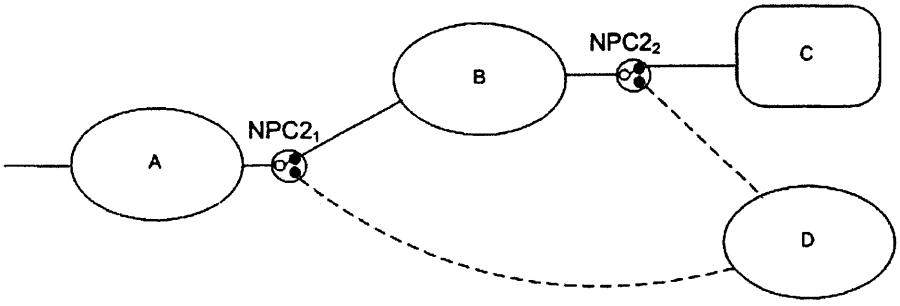


Figure 54: Use of multiple NPC2 constructs

In the connectivity matrix C_{11}, the NPC2$_n$ is represented as b_n where the conditional connection is active and $\overline{b}_n$ when inactive. C_{11} in (39) shows the configuration where the domains are represented as $\mathbf{D}_{11}$ = [A, B, C, D].

$$C_{11} = \begin{bmatrix} 0 & b_1 & 0 & \overline{b}_1 \\ b_1 & 0 & b_2 & \overline{b}_2 \\ 0 & b_2 & 0 & 0 \\ \overline{b}_1 & \overline{b}_2 & 0 & 0 \end{bmatrix} \quad \ldots (39)$$

The total is calculated in (40) giving 4 potential compromise paths for the case $b_1 = 1$, $b_2 = 1$, $\overline{b}_1 = 0$ and $\overline{b}_2 = 0$. Note that each instance of b_n and $\overline{b}_n$ can have the value of 1 or 0. There are three other cases to consider although the total number of paths would be the same in each instance.

$$Total = 2b_1 + 2\overline{b}_1 + 2b_2 + 2\overline{b}_2 = 4 \quad \ldots (40)$$

Here, we have used the Boolean identities $b_1 + \overline{b}_1 = 1$ and $b_2 + \overline{b}_2 = 1$.

What is interesting about this approach is the flexibility it offers to model numerous different configurations within a *single* matrix. For example, in Figure 54 there is no relationship between the two NPC2 constructs therefore it would be necessary to calculate the matrix 4 times in order to capture the 4 combinations of NPC2 state. This can easily be achieved through the single model simply by substituting the

appropriate value of b_n into the NPC2 of choice. A different security policy may specify that NPC2s must operate together, that is, their individual behaviours are effectively linked together. If the conditional connector is activated on one, another within the same chain is also activated. It is possible to capture this event within the matrices because the NPC2 can share the same identifier. Figure 55 illustrates this scenario where when activated, both NPC2 constructs in the model will switch to permit the alternate business connection in the model. The effect on the matrix C_{12} is shown in (41).

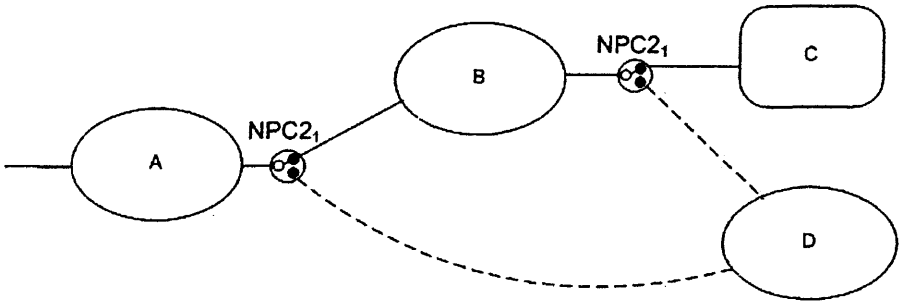


Figure 55: Multiple NPC2 constructs operating as a chain

$$C_{12} = \begin{bmatrix} 0 & b_1 & 0 & \overline{b_1} \\ b_1 & 0 & b_1 & \overline{b_1} \\ 0 & b_1 & 0 & 0 \\ \overline{b_1} & \overline{b_1} & 0 & 0 \end{bmatrix} \quad \ldots (41)$$

As shown in (42), the total number of potential compromise paths is 4 when $b_1 = 1$ and $\overline{b_1} = 0$.

$$Total = 4b_1 + 4\overline{b_1} = 4 \quad \ldots (42)$$

In this configuration, if the position of the NPC2 were to be switched, the values of b_1 and $\overline{b_1}$ would be transposed that is, $b_1 = 0$ and $\overline{b_1} = 1$, which would not alter the total number of compromise paths. Both cases are covered by a single equation, (42).

The results in (40) and (42) show that total connectivity is the same for the two configurations despite the fact that the NPC2 is being used differently. This is somewhat counterintuitive; our initial expectation was that when the NPC2s are

linked together total connectivity would be less than when the NPC2s operate independently. Figures 54 and 55 do represent a simple case. Therefore in order to explore this phenomenon more closely another example is given in Figure 56.

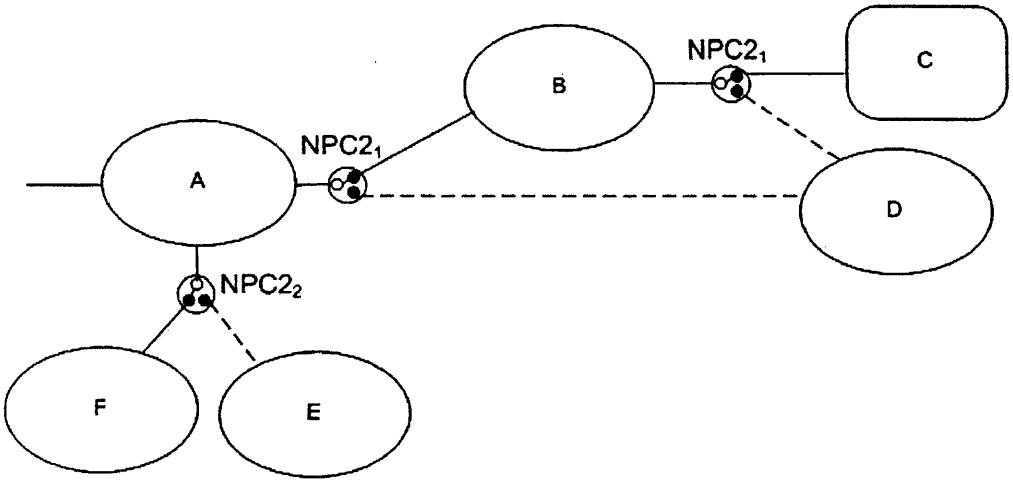


Figure 56: Multiple NPC2 constructs working independently and as a chain

Figure 56 has 3 NPC2 constructs. Two of them, labelled as $NPC2_1$, operate together whilst $NPC2_2$ operates independently. The domains in Figure 56 are represented as $\mathbf{D}_{13}$ = [A, B, C, D, E, F]. Connectivity for the cases $b_1 = 1$, $b_2 = 1$, $\bar{b}_1 = 0$ and $\bar{b}_2 = 0$ is given in the matrix C_{13} (43). As with the earlier examples, if the position of any NPC2 is switched the values of b_n and $\bar{b}_n$ would be transposed.

$$c_{13} = \begin{bmatrix} 0 & b_1 & 0 & \bar{b}_1 & \bar{b}_2 & b_2 \\ b_1 & 0 & b_1 & \bar{b}_1 & 0 & 0 \\ 0 & b_1 & 0 & 0 & 0 & 0 \\ \bar{b}_1 & \bar{b}_1 & 0 & 0 & 0 & 0 \\ \bar{b}_2 & 0 & 0 & 0 & 0 & 0 \\ b_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \ldots (43)$$

Total connectivity for this configuration is calculated in (44) where it can be seen that a total of 6 potential compromise paths exist.

$$Total = 4b_1 + 4\bar{b}_1 + 2b_2 + 2\bar{b}_2 = 6 \quad \ldots (44)$$

To conclude this example, each of the NPC2s in Figure 56 are considered as independent entities. This case is shown in Figure 57.

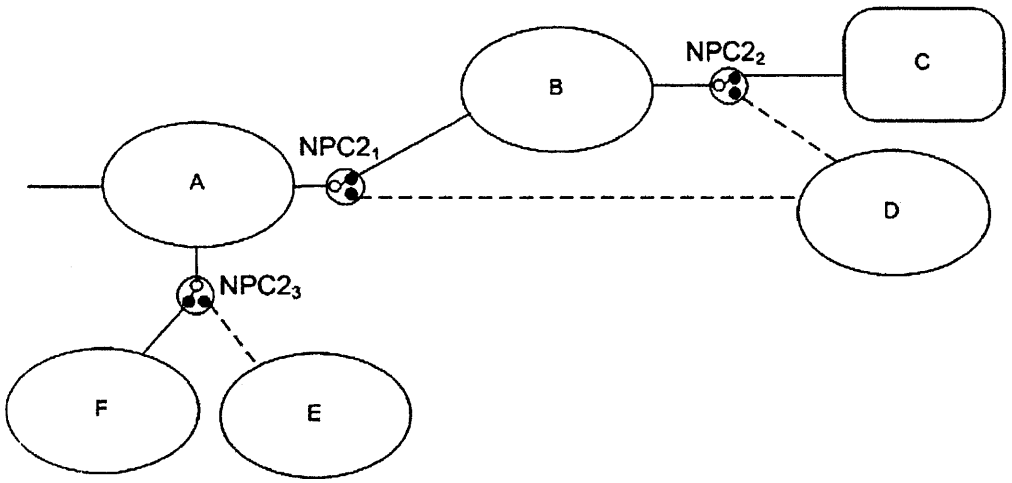


Figure 57: Multiple NPC2s operating independently

The domains in Figure 57 are represented as $\mathbf{D}_{14}$ = [A, B, C, D, E, F]. Connectivity is shown in C_{14} (45).

$$C_{14} = \begin{bmatrix} 0 & b_1 & 0 & \overline{b}_1 & \overline{b}_3 & b_3 \\ b_1 & 0 & \overline{b}_2 & b_2 & 0 & 0 \\ 0 & \overline{b}_2 & 0 & 0 & 0 & 0 \\ \overline{b}_1 & b_2 & 0 & 0 & 0 & 0 \\ \overline{b}_3 & 0 & 0 & 0 & 0 & 0 \\ b_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \ldots (45)$$

Total connectivity is calculated in (46) and yields 6 potential compromise paths.

$$Total = 2b_1 + 2\overline{b}_1 + 2b_2 + 2\overline{b}_2 + 2b_3 + 2\overline{b}_3 = 6 \quad \ldots (46)$$

Total connectivity in (46) is the same as (44) even though the NPC2s in Figure 56 are acting independently. This more complex configuration gives the same result as the simple examples in Figures 54 and 55, that is, there is no difference in the total number of compromise paths irrespective of whether the NPC2s operate independently or as a chain. The results obtained in (40) and (42) and (44) and (46) lead us to the following observation:

In a network exhibiting symmetrical connectivity, the total number of connections will be twice the number of NPC2 constructs used to control connectivity irrespective of whether the NPC2s act independently or in a coordinated fashion.

This observation is clarified as follows. The Boolean variables $b_1, \overline{b}_1, b_2, \overline{b}_2 \ldots$etc occur in pairs; that is, for every occurrence of b in the matrix there is a complement $\overline{b}$. The NPC2 offers 2 potential paths, symbolised by b and $\overline{b}$. In examples such as (46) they always occur as a sum $b+\overline{b}$, and this is, by definition, unity: $b+\overline{b}=1$. In a symmetric matrix each NPC2 offers 2 bi-directional connections therefore to obtain total connectivity, the number of connections are doubled. Our observation is useful since in appropriate configurations it can be used as a shortcut to total the connections and thus the number of potential compromise paths. Furthermore, it provides a means of verifying the addition of the matrices. Understanding the relationship between NPC2 connectors enables us to verify the matrices. Since all of the Boolean values should have a complement, mistakes can be readily identified through a visual check of the matrix. This theorem has not been tested against an asymmetric case and we do have doubts concerning its applicability in this circumstance. Such a study would be of interest but is beyond the scope of this thesis. In summary, the use of a formal technique, such as the matrices, is necessary to reveal subtle and counterintuitive results. It would not have been obvious from a simple visual check of the models, that the number of compromise paths would have been the same irrespective of whether multiple NPC2s operate independently or as a group.

4.6 Discussion of Preliminary Results

The preliminary results regarding indirect connectivity and network growth can be discussed in the context of other work. Metcalfe's law [45] as stated in (1) applied to a 6 node network yields a total of 15 connections. If an additional node were added this number would increase to 21; a substantial increase in connectivity. Our models assume bi-directional connectivity. Therefore these values would actually be 30 or 42 potential connections respectively. Given that our work equates *connectivity* with *potential compromise path* if Metcalfe's calculations are correct, additional

connectivity would probably result in an unacceptable increase in risk. However, Metcalfe makes the assumption that nodes are fully connected therefore when an additional node is added it is automatically able to connect to all existing members (apart from itself). The models in Sections 4.2 and 4.5 showed the difference in potential compromise paths when a new domain was both fully and partially connected to existing domains. The result was quite pronounced and can be further influenced through the use of the NPC2 and NPD constructs.

Our initial findings indicate that the rate of growth does not conform to Metcalfe's model neither is it as pessimistic as the view of maximum permitted connectivity. Taking Figure 21 as an example we apply Metcalfe's formula to the model. Since there are three domains this would take the form $3(3-1)/2$ and would yield 3 connections. Our results for Figure 21 showed a total of 4 potential compromise paths. This is clearly a larger value when compared with Metcalfe's value of 3 but is explained by the fact that unlike Metcalfe, each connection is considered independently and the result contains no division by 2. Our result is still less than maximum permitted connectivity (6 connections) as shown in Figure 20, which does not recognise the role of the NPC2. These preliminary results indicate that the NPC2 and NPD constructs have a positive impact on reducing the number of potential compromise paths. Moreover, their use can be effective in controlling the rate of growth and proliferation of connections as domains are added to a network. This subject has by no means been exhaustively researched in this thesis. We conclude that further work is required in order to wholly appreciate the implications of including non-persistent and conditional constructs in the risk assessment process.

Chapter 5: Implementation of Fit for Purpose Security

This chapter is the first of two which together represent a paradigm shift in our work. They describe the transition from a conceptual view of the NPC2 in high-level models to practical implementation of the principles in a small-scale example.

Chapter 5 presents the evolution of the NPC2 from a piece of abstract modelling notation used in Chapters 3 and 4 to an architectural model and design blueprint. The model identifies how the NPC2 will operate in and interact with its target environment and the components necessary to achieve the required behaviour. Since our overall objective is to utilise the NPC2 in high assurance environments, we were bound to observe certain design objectives which will enhance its acceptability into our target market. In following this route we became aware of the potential for designing a proprietary framework which could result in a 'closed' solution completely at odds with our philosophy of fit for purpose security. The level of openness in the NPC2 model can be visualised as an up-side-down pyramid such as that shown later in Figure 59, where the modules at the top are generic and platform-independent. As the architecture progresses towards the security policy components so it becomes more focused and constrained by the target platform. This is a concept we return to in section 5.1.

In Chapter 6 we describe the practical element of our work where the NPC2 design has been taken and engineered into a working example. This prototype solution is one way in which the NPC2 can be constructed; there are of course many others which could utilise different technologies. A major strength with our approach is the openness of the NPC2 structure which means that systems engineer may utilise their preference of technology provided the overall design principles are adhered to.

5.1 Visualising the Non-Persistent Capability Concept

In the high level modelling process the NPC2 effectively provides an indication that a level of flexibility is required through an exception to the normal or baseline separation and sharing requirements. Until the risk analysis stage where security

controls are considered, we are not concerned with *how* flexibility will be engineered into the system. As described in Chapter 1, fit for purpose security is underpinned by the synthesis of security policy, architecture and mechanism. As a result, a substantial part of our background research focussed on looking at the separation architectures and technologies that were available and would ultimately influence the design of any fit for purpose solution we would propose. In principle the NPC2 sits above and manipulates the underlying system to elicit changing behaviour in accordance with a security policy. The NPC2 therefore needs to be as generic as possible so it can be applied across a variety of different platforms and technologies with minimal reconfiguration. Realistically, we appreciate that at some point in the design it will be necessary to bind the NPC2 more closely with the underlying technology and lose some of its open structure. The NPC2 acts as a 'glue' between the external environment comprising other host systems and the system platform on which it is itself hosted. The host platform is expected to support some form of separation between objects although the strength and capability of this mechanism would of course vary depending on the solution employed. The NPC2 does not mandate any particular type of target technology although it will require a certain degree of customisation in order to interface with individual security sub-systems. In summary, we have applied some of our early research on separation architectures to the design principles and development of the NPC2 architecture. These include:

- modularity of components
- clearly defined and bounded functionality
- technology independence

During the course of this chapter it will become clear how each of these principles has been utilised in the framework and where difficulties have been encountered. We discovered that moving from statements of intent, which constitute the design, through to a practical implementation is a complex task, particularly when we are seeking to introduce an element of dynamism into the system. As shown in this section, the NPC2 is not simply about a security policy that can be defined, refined and processed into system level rules. In order to adapt to changes in the environment in an appropriate way, the policy must interact with other components that observe

and monitor conditions, determine when a threshold has been met or exceeded and subsequently invoke a change in the security policy. Each of these processes could be carried out by a 'human in the loop'. Given that 'system robustness', in accordance with Ashby's Law of Requisite Variety [5], and fit for purpose security are both design goals for the NPC2, our solution is designed to operate autonomously. The policy is therefore wrapped within a suite of supporting capabilities, including:

- Environment monitoring / detection
- Event management including policy trigger mechanisms
- Validation

In accordance with our design objectives, the monitoring, management and policy requirements which embody the NPC2 have been refined into a series of modules. The result is a package of individual but integrated components, which when combined provide a practical means of realising the NPC2. Figure 58 presents a conceptual view of the NPC2 architecture.

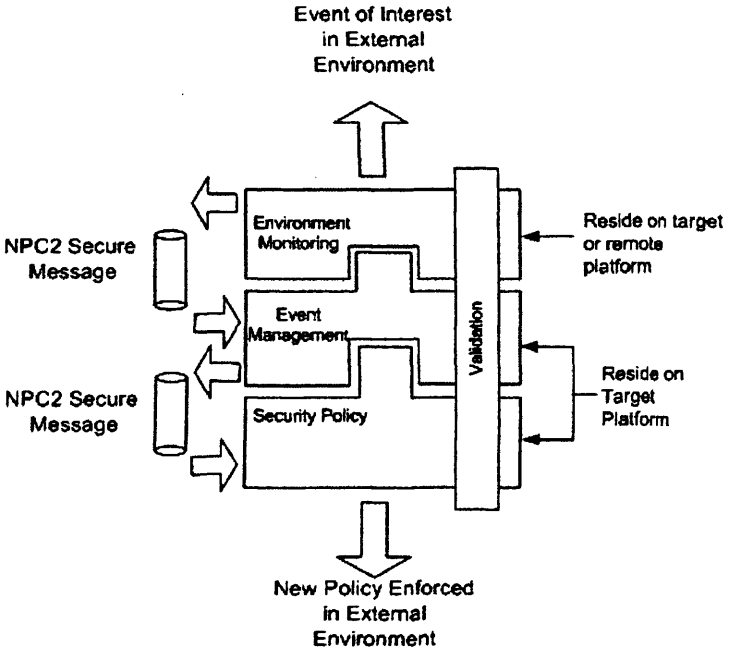


Figure 58: High level taxonomy of NPC2 architecture

Each of the elements in Figure 58 are described in later sections of this chapter. However, the following points are noteworthy at this stage since they will

significantly affect the design and implementation and are fundamental to our approach.

Given that the NPC2 interacts with and controls a host platform, there will be some overlap between functionality at the lower layers of the architecture and the host security sub-systems. We expect the policy trigger mechanism (which is considered a part of the event management module) to be more closely coupled and therefore potentially less generic than other components of the NPC2. A conceptual view is shown in Figure 59.

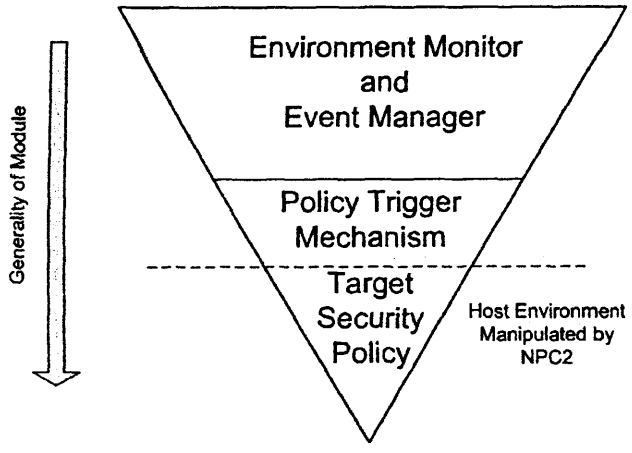


Figure 59: Conceptual view of NPC2

Both Figures 58 and 59 identify the relationship between the NPC2 and the host environment which it influences by controlling changes to the target security policy. This environment of course is external to the NPC2 itself and is therefore technology specific to the host platform.

Figure 58 also identifies the direction of information flow between components and it is immediately clear that the environmental monitoring component is both inward and outward facing. That is, it collects data (externally) from the environment and feeds it to an internal store. This is noteworthy because the fit for purpose security principle is dependent, at least in part, on the trustworthiness of its components. Quite clearly, the external environment presents a risk to the system. Therefore the veracity of the data collected must be confirmed as far as possible. Furthermore, a security mechanism must be applied to the internal communication process to protect data

being passed between components. The same principles will apply to other communication flows identified in Figure 58.

We have identified where components of the NPC2 could potentially reside in the target system. That is, they could be installed locally on the device where the security policy will be updated, or distributed within the system. Of course the placement of components will be driven by the implementation scenario since it will influence the development and functionality of software modules. The demonstration environment used for our example in Chapter 6 monitors the absence of a device, and in order to detect its presence or absence our monitoring event is bound to the network. For simplicity the Environment Monitor module is activated on the device that will assume command. Equally, this capability could have been distributed within the network and collected information from other community members. It would have introduced other complexities and issues for the security of the system, although there would have been other benefits. For example, distribution of the capability removes the problem of a single point of failure which we encounter if monitoring is contained within a single device. Furthermore, our device may simply be out of range of the Commander device and therefore erroneously report its absence. If other devices are also responsible for collecting the information then it would be apparent that the 2IC was at fault and not the Commander device. These are issues we are aware of. The design of the NPC2 therefore takes account of both local and remote data collection. The choice of approach can only be decided in the light of other information related to the individual scenario.

Validation is a feature of all components and of course the entire NPC2 will require verification that it functions as designed. Given our interest in high assurance environments the ability to validate is a critical part of the architecture. The modular approach of the NPC2 assists with this process since each component is small and very self-contained. The interfaces between an individual module and its neighbours are well defined and bounded. Of course the complexity of each module will increase as additional functionality is included to perform more complex event processing or collection of multiple events. The benefits of enhancing a module must always be weighed against the impact on validation.

The NPC2 does not stand as an independent entity within the system. It is required to interface with the existing environment and indeed, some aspects of it will actually straddle the boundaries between the NPC2 and external environment. As described previously, one example is the security policy. This clearly exists in the external environment but is manipulated by the NPC2 to achieve flexible behaviour. Although we alluded to the presence of the external environment in Figure 58, the architecture framework itself did not provide any contextual information to describe how the NPC2 fits within an overall configuration. This detail is captured in Figure 60.

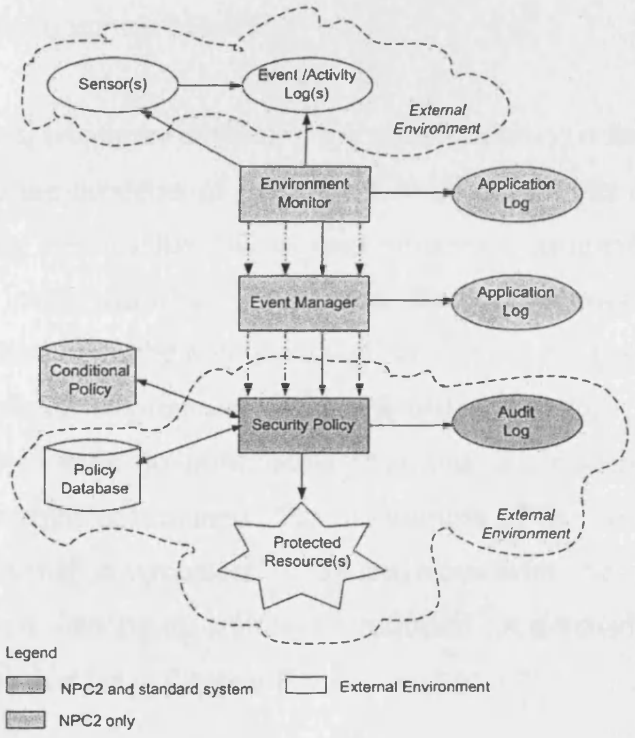


Figure 60: NPC2 in context

Figure 60 distinguishes between elements that will be developed to form the NPC2 only and components that will be modified or adapted from the standard (target system) to provide or enhance NPC2 capability. Further information on each module is provided in the sub-sections which follow.

5.1.1 The External Environment

Figure 60 shows the NPC2 positioned within the external environment. This represents the elements used to drive policy change, because they are either events of

interest or elements that will be affected by a change in policy rules. The external environment is not prescriptive; that is, we could register an interest, through the Environment Monitor component for an event or change in time. The events themselves may be interpreted in many different ways. For example if communication appears from a specific source we may perceive the event as an increase in risk or a reduction in trust and thus wish to change our separation and sharing profile accordingly. Figure 60 indicates that environmental data can be gathered from external sensors directly and fed straight into our Environment Monitor component or it could be an indirect feed collected from a log file that is subsequently monitored by the Environment Monitor.

Access control rules, which are defined in the security policy, reflect the overarching separation and sharing concerns of the system stakeholders. In our case study for example, the policy permits the 2IC to read situational awareness data when the Commander is no longer available. This action effectively allows the 2IC to 'share' this information where he or she was 'separated' from it before. These access rules are enforced in the external environment and manipulated by NPC2 functionality. The high level models provide no information regarding the resource(s) that will be protected in the external environment. For the purpose of our research it should be noted that the 'external environment' is synonymous with the 'test environment' where the NPC2 as a working capability is scrutinised. A detailed description of the test environment is provided in Chapter 6.

5.1.2 Monitoring the Environment

It is envisaged that a system will be deployed with a baseline set of separation and sharing conditions that fulfil the anticipated requirements of the scenario. These are captured in the high level modelling process where in the DBSy InfoSec Architecture model, presented in earlier chapters, solid lines or permitted business connections join domains together to indicate that sharing is permissible. In order to allow the system to adapt to change it is necessary to have some means of detecting 'events of interest' in the environment that will ultimately drive through a change in security policy. In the NPC2 architecture this component is the Environment Monitor which we expect to be implemented through software or in some situations a combination of hardware

and software depending on the installation and application. The Environment Monitor consists of two elements, the event monitor itself and a repository, which are shown in Figure 61.

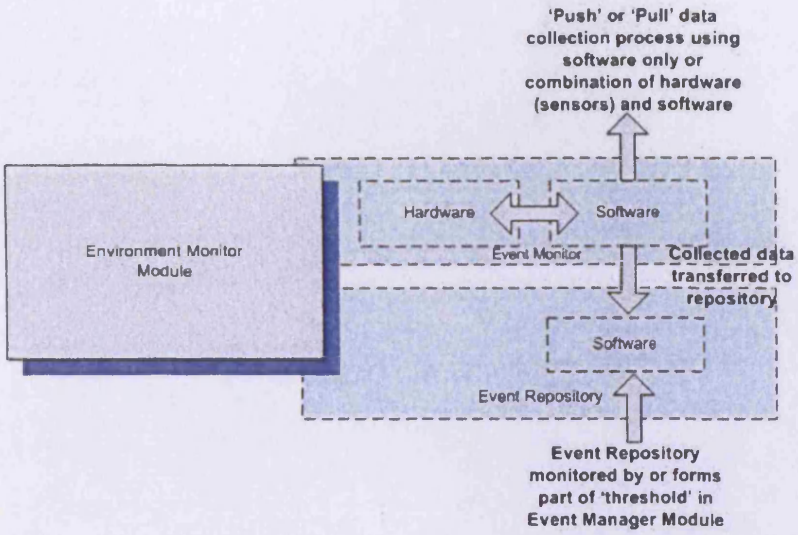


Figure 61: Conceptual view of Environment Monitor Module

As shown in Figure 61 the Environment Monitor module essentially provides a data collection service where data related to the 'event of interest' is collected and fed into a temporary repository for further processing. The monitoring service could be implemented as a combination of hardware and software or software only depending on the type of event being monitored. For example, one solution may include the use of hardware sensors with onboard software designed to detect the presence of moving metal objects such as vehicles. The event of interest could be detection of the object in an area identified as 'no access' which would subsequently generate and send a message elsewhere in the system. Interestingly the NPC2 Environment Monitor in this example could be the actual sensor and software itself or it could be an independent software process monitoring the generated message through the temporary repository or receiving the detection message. In Figure 61 it is shown that the data collection process may be 'push' or 'pull'. In other words, the Environment Monitor may pull data about the event of interest from the detection process. It could also subscribe to the same process and be informed about the event of interest when it occurs. The method employed and actual composition of the monitor would depend entirely on the requirements of the target system.

In Figure 60 multiple lines were identified connecting the Environment Monitor to the Event Manager. This indicates that the Environment Monitor may be interested in and therefore passing on information about one or possibly many events from the external environment. Multiple configurations are envisaged. It could be configured to detect multiple instances of the same event or perhaps single instances of different events. This would enable scalability in the NPC2.

5.1.3 Event Manager

Figure 62 is conceptual view of the Event Manager architecture. This can be implemented as a software component and is comprised of two components; the threshold which when met or exceeded generates an event or alert of some description, and the Event Manager itself which is responsible for taking appropriate action on receipt of the alert communication. This action could be monitoring and maintaining the status of the environment variable that has been set.

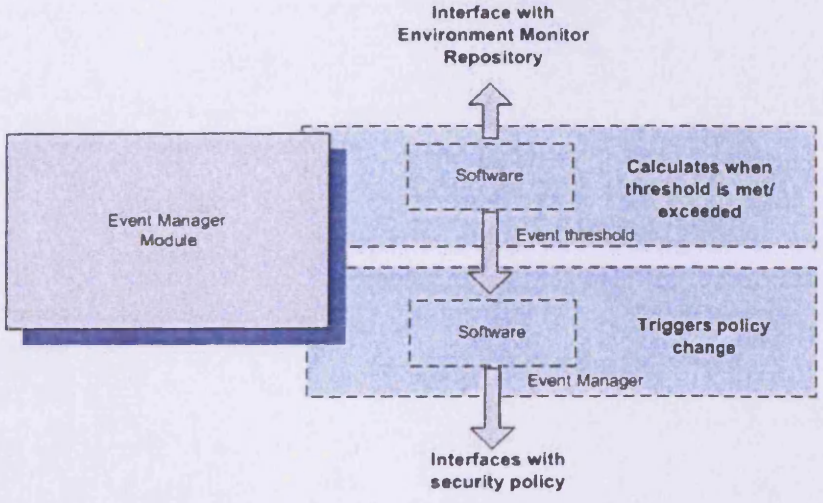


Figure 62: Conceptual view of Event Manager Module

Essentially we want to cause a change in capability which involves loading a modified set of rules into the system either as a complete policy or module. The Event Manager will include a software component that is responsible for triggering a change in the security policy. The physical operation of the trigger mechanism will be dependent on the capability of the target environment. For example, the demonstration system created and described in Chapter 6 is based upon SELinux. It is therefore possible to have a minimal implementation for the trigger mechanism and

harness the conditional policy extensions through manipulation of a Boolean variable; both of which are features of SELinux. Another security infrastructure which is less flexible than SELinux would require a different approach.

The Event Manager can be as simple or complex as the situation demands. For example, if only one input is coming from the Environment Monitor then the Event Manager needs only to manage and process one interrupt. However, if it is receiving multiple pieces of data then some level of additional processing is necessary. This may involve:

- fusing the environment data to generate one trigger (many to one)
- fusing the environment data to generate multiple triggers (many to many)
- managing events where there may be one or more events and/or triggers but the relationship between them is one to one
- some combination of the above

Figure 60 also captured this possibility by showing multiple lines between the Event Manager and Security Policy.

5.1.4 Security Policy

The security policy is a difficult module to describe. It clearly has a significant role to play in the NPC2, but fundamentally belongs to the external target platform and is therefore *required* to interface with the NPC2. For the purposes of this chapter it is necessary to present the properties that are required from the security policy whilst avoiding details that would be specific to the security sub-system of a target device. The properties described will almost inevitably dictate the type of policy which can be applied on the target system. It is necessary, for example, for the security policy to support dynamic behaviour. This behaviour is characterised by the loading and unloading of rules that will enable or revoke access with minimal disruption to the end user. At this point our vision of fit for purpose security becomes very contentious with systems operating at the higher end of the evaluation assurance levels and enforcing mandatory access control. Typically the security policy loaded at system start will not change once the system is operational. If a change is required the system

must be restarted. Quite clearly this does not fit with our vision particularly where the affected system is operating in an autonomous environment. Nonetheless, our whole argument is based around the need for a more radical view of security and we believe there is sufficient motivation to consider dynamic security policy but under controlled circumstances. The level of dynamic capability supported in the security policy will without doubt influence the development of other modules in the NPC2. For example, our research revealed that SELinux supported conditional policy extensions triggered by a change in the Boolean variable. As shown in Chapter 6, this made the development of the Event Manager, which encompasses the trigger mechanism, very easy. If this capability had not been available, engineering of the same mechanism would have been more complex because it would have had to perform some of the tasks undertaken by the conditional policy rules themselves.

It is suggested that the NPC2 solution has some form of policy feedback mechanism which is invoked after a change to the policy is loaded. We foresee a requirement that once a modified policy has been activated it may be necessary to inform the affected 'user' of the change. In situations such as our own the tempo of operations is such that it is essential for the user to begin applying the new policy immediately. It would be inappropriate to wait until the next time the user makes an access request. This approach differs from the traditional view of the policy decision and enforcement process provided by the reference monitor [30]. Normally a user requests a capability which is checked and permitted (or denied) by the reference monitor. The whole process is thus user driven. It should be noted, that under this regime the user will be unaware of a policy change until their access is refused and they probably will not know why it has failed. Worse still, if a user has never had access before but the policy changes to allow a capability, they will be ignorant of that fact unless a feedback mechanism is in place. In a more dynamic approach we expect a level of 'planned intelligence' where some decision can be taken regarding updating affected parties that the policy has been changed.

5.2 Validation

As stated previously, validation is a significant element in the design of the NPC2 and therefore appears at all levels of the architecture. Validating the design and implementation of the NPC2 modules can be achieved in a variety of ways and

depending on the requirements of the stakeholders, some aspects will be more important or useful than others. The purpose of this section is to describe some of the features both in the NPC2 design and the way in which we anticipate it will be constructed that will contribute towards validation. The section does not discuss specific technologies that provide any of the features described since it is our intention to maintain independence in this respect.

It is recognised that where an organisation ultimately seeks a level of assurance for a product it will be necessary to follow a more structured design and development process right from the start. However, for the purposes of this research it was not appropriate to conform to such requirements because the work is a proof of concept rather than a finished product. Nonetheless, we are mindful of this possibility in the future. Although it is clearly an area for future work, which we contend would come under the umbrella of development as opposed to research, as part of our discussion on validation in both this section and within Chapter 6 we present some of the features of the NPC2 design which should assist with an evaluation. Our aspiration is to keep the core NPC2 components as small and simple as possible with targeted functionality. This philosophy is in harmony with established guidelines and architectures for secure systems. For example, in [28, p.140] Anderson notes how the study by James Anderson [28]:

"...led the US government to conclude that a secure system should do one or two things well; and these protection properties should be enforced by mechanisms that were simple enough to verify and that would change only rarely..."

We examine the design of the NPC2 in the context of this description in Section 5.2.1. The NPC2 is modular and we have attempted to clearly define and bound the functionality provided by each of the layers. In theory it should be possible to implement the NPC2 with small modular applications working together to provide the overall capability; applications which should be more amenable to testing to verify for accuracy, integrity and functionality.

Validation is not restricted to the development of each NPC2 module but in the communication and information exchange between modules and of course the

underlying operating system that supports it. In Figure 58 we show secure messaging as a requirement between each of the modules. Again, the interpretation and implementation of such a feature would be governed by the needs of the stakeholder and the capabilities of the technology used for implementation. We look at the NPC2 in the light of other architectures in Section 5.2.2.

Logging and auditing perform a significant role in the validation process which is discussed in Section 5.2.3. It is necessary to have some record of events that can be easily analysed and potentially used as a feedback loop to assist with system refinement. We would therefore expect logging and auditing to be an essential part of the initial development to confirm that the system behaves as expected and validate both the design of individual components and the overall capability.

Finally, it should be noted that since the NPC2 is technology independent, the final solution including the target platform hosting the NPC2, could be constructed using security certified components. Although this does not guarantee a level of certification, because the resulting system would need to be re-certified, it would clearly assist in the evaluation and certification process.

5.2.1 Design Principles

The NPC2 architecture is modular by design and presents a number of advantages. The software components within each module perform very specific tasks which should enable the developer to contain the size and complexity of each. A modular design also assists with application testing and the validation of software code. For the purposes of evaluation it is necessary for the independent assessor to examine individual lines of code to verify the integrity of the software. Modern operating systems and applications are victims of their own functionality. The enhancements to functionality result in an increase in the amount and complexity of code required to deliver the capabilities. Unavoidably the software will become more expensive and difficult to evaluate. We have an opportunity to design the total NPC2 capability around small composable modules. Therefore, in theory it should be easier to confirm the behaviour of each module independently and as part of the whole structure.

A modular design also allows us to secure each application independently and contain its access to resources. Since the NPC2 modules are well defined and specific, greater control can be exerted over the access required for the individual applications to perform their task. This design allows us to adopt the *principle of least privilege* [65]. In the event that the application is flawed, there is some potential to contain damage it may cause to other processes running on the system. Of course, this level of control will be dependent on the underlying functionality of the host security system. If it does not provide the level of granularity required to secure individual applications, then it is not possible to exercise strict control over the NPC2 modules. As part of the fit for purpose vision, it would therefore be necessary to select a host operating system that offered flexible security for the NPC2 components themselves in addition to supporting the capabilities required to support dynamic changes to the security policy rules.

The modular design is of course advantageous when performing functionality tests since each can be tested independently of another. Furthermore, it is possible to perform incremental testing where modules are gradually slotted together and tested until the entire system is constructed.

5.2.2 Security of Messages

Validation and the security of messages relates to ensuring the veracity and integrity of the message communicated between different components of the NPC2 and in some cases the NPC2 and the external environment. Of course this will be dependent on the level of communication. That is, we are potentially dealing with messaging between devices using the communications network and messages passed between processes on the same system. In either case we need to be sure that the message can be trusted since it will ultimately be used to drive through a change in the security profile of the system. There are of course different techniques for validating the authenticity and integrity of a message which involve the use of encryption and digital signatures. At this point it should be remembered that the NPC2 may be used in autonomous systems with limited resources therefore the overhead of applying cryptographic techniques must be carefully reviewed. The MILS architecture [35] outlines the use of end-to-end encryption for inter-process communication and since

the architecture has successfully been implemented in small embedded systems such as those used in the avionics industry, [36] we know it is achievable in a constrained environment. The ability to provide validation through the security of messages communicated to and within the NPC2 will clearly be dependent on the capabilities of the implementation platform. In Chapter 6 we discuss how this aspect was approached for the demonstration system.

5.2.3 Logging and Auditing

The NPC2 components generate application logs where their activity will be recorded in the form of an audit log. The logs can be analysed to provide some confirmation that the component is behaving as expected. It is anticipated that the technology adopted for the security sub-system will also provide some form of audit log where access attempts are recorded. The NPC2 modules will be protected by a security policy and their own access attempts will be subject to auditing. Again, this will enable the system analyst to confirm that the component is not attempting to access a resource for which it is unauthorised. If it does, the attempt can at least be validated to determine whether it is legitimate or not and appropriate action taken.

5.3 Summary

This chapter describes the transition from the NPC2 as a modelling capability to the NPC2 as an implementation architecture. The purpose was not to prescribe a set of technologies that should be adopted, but outline the functionality and interactions between the suites of components deemed to be necessary to achieve a flexible security capability.

Chapter 6: Practical Demonstration

This chapter presents the practical phase of the NPC2 implementation and the development of our prototype solution from the architecture described in Chapter 5. As with most engineering projects the actual implementation differed from the theoretical model. However this is a valuable part of the process since it is necessary to establish the extent to which the theoretical concepts map to reality. Of course an account of the 'lessons learned' from this experience is useful in future implementations both for our own purposes of that of other developers. In the sub-sections that follow, we describe how each component of the NPC2 was engineered for the demonstration system. This solution is, of course, only one example of how the NPC2 might be applied in practical terms and has been developed specifically for our requirements. The example described is intended to be a proof of concept and would therefore require more rigorous engineering to evolve it from prototype to product. At this stage the development process would be aligned with the requirements of a formal evaluation model if a level of assurance was necessary for the NPC2.

6.1 Development Considerations and Constraints

The transition from the design and modelling phase to practical implementation revealed a number of issues. These were external to the NPC2 but nonetheless influenced the way in which the work evolved and the solution was engineered. This section describes each and discusses how the issue was addressed in the development. This is introduced via a reprise of the original case study since its content raised some unique challenges to be overcome.

6.1.1 The Scenario

The demonstration scenario is based upon an agile group and its interaction with other parties, who by virtue of their affiliation and function, have different levels of trust associated with them. This situation has been captured in the DBSy InfoSec Architecture models presented in 3.3.1 of this thesis.

In our scenario, Domains AX, AY and B are all members of an agile mission group Domain AX is permitted to share information with Domain AY using messaging and Domain AY shares information with Domain B. Members of Domain AX are not normally permitted to share information with Domain B. However, under exceptional and particular conditions which, in this case occurs when Domain AY is unavailable, sharing between Domains AX and B is permitted. At this time there is no connectivity between Domains AX and AY. The absence of Domain AY must be detected so that the security policy can be modified to allow the system to adapt accordingly. As described in Chapter 3, our work includes the NPC2 to capture temporary and conditional connectivity within the DBSy InfoSec models.

In the models, the NPC2 is directly connected to the domain where it will be implemented; although at this level of detail there is no indication regarding either the platform or technology that will be used to realise the NPC2. That said, the NPC2 quite clearly controls the ability to pass or share information between two domains and this level of control could manifest itself in a number of ways. For example, we could adopt the typical functionality of a gateway device where members of the same community or network are permitted to communicate with relative freedom. Traffic crossing the gateway is subject to restrictions. The desired levels of separation are achieved by blocking communications packets at the network boundary based upon factors such as the source or destination address of the packet, protocol type or possibly the port address. This is a classic gateway approach but does present some risk since a significant part of the separation strategy is placed in a boundary device that might fail. It is usual to design 'defence in depth' where a firewall or gateway is combined with other security controls. Details such as these are identified and assessed as part of the risk analysis process.

Another approach is to define the separation boundary at a lower level. For example, members of a domain could be permitted or denied the capability that enables them to share information with a different domain. Whilst this affords greater granularity of control, that is, it is possible to specify who has elevated privileges, we also risk rule explosion and greater complexity if there are too many exceptions to the 'normal' security policy. However, this second approach is in the spirit of our original concept for the NPC2. We recall the literature review in Chapter 2, where the divergence

between our thinking and the principle of *weak tranquillity* was observed. An advantage with our fit for purpose solution was found in its ability to tightly control an elevation of privilege and open a system for a limited period, to a limited number of subjects for limited capabilities. It is therefore necessary to design our implementation to uphold this philosophy. A separation solution using the NPC2 exercises control over *when* a user may acquire enhanced privileges in the system and thus modifies access to the flow of information accordingly. The NPC2 enables the *principle of least privilege* [65] to be designed into a device that must also provide a level of flexibility.

6.1.2 Engineering Resources and Skill Sets

As stated previously, the NPC2 is comprised of different components including hardware (possibly), software and security policy. It was apparent from an early stage that a blend of skills would be required in the practical experimentation phase to support the development process. Broadly speaking, these skills cover software development or programming expertise and specific knowledge of the selected security policy language or environment used in the implementation. It was unrealistic to assume that one person would necessarily possess all of the skills required to the necessary level of expertise. Furthermore, given the timescales of the project it was impractical for one person to acquire any additional skills. Therefore we took the approach that collaboration and specialisation in the defined areas presented a satisfactory solution to the issue. The experimentation was therefore undertaken by the author and a software developer, a colleague, who was also familiar with the target platform and critically, the philosophy behind the NPC2.

Our collaborative approach brought with it additional considerations. The most significant for this research was a modification to the first example of the NPC2 where it was necessary to demonstrate its capability in a more visible and therefore acceptable format to the stakeholder providing the development resource. Of course, security policy operates in the background and is therefore neither visual nor obvious to the casual observer. An effective way of demonstrating a working security policy is to show the effect by denying a capability followed by the difference in behaviour when the same capability is permitted. The 'effect' of course needs to be visual to

make a satisfactory impact. In our situation it was necessary to adapt the interpretation of our original case study to make use of display output. The modifications allowed us to maintain the fidelity of the original principles, that is, separation between lower and higher domains unless exceptional conditions arise, whilst producing a very visible example to show the NPC2 in operation. The revised scenario is as follows:

1. Situational awareness data, in the form of Global Positioning System (GPS) co-ordinates, time and user identity is distributed to all users of the network and written to a message queue.
2. Device 1 (the Commander) is permitted to read the message queue. The situational awareness data is read, interpreted and displayed on a hand-held output device to show the geographical location of the user on a map. This updates as the device changes position.
3. Device 2 (the Second in Command or 2IC) is not *normally* permitted to read the message queue. The messages are therefore discarded[1]. The display on the output device is inactive because no situational awareness data is available.
4. Security policy exception - in the event that Device 1 is unavailable[2], the security policy changes to permit Device 2 to read the message queue. Situational awareness data is read, interpreted and displayed on the output device and updates as the device changes position. The output display for Device 1 becomes inactive since it is no longer participating in the network.

Figure 63 illustrates stages 1-4 from the list.

---

[1] This is a recognised vulnerability in the design. If the security policy is undermined, the device will be able to read messages. However, the ad-hoc network is a broadcast network therefore all devices will receive the situational awareness messages.

[2] The term 'unavailable' may be interpreted in many ways. It is necessary to differentiate between the human operator (who may be incapacitated when the device is functioning) and the device itself which may simply malfunction. For this demonstration it is assumed that the device itself will become unavailable.

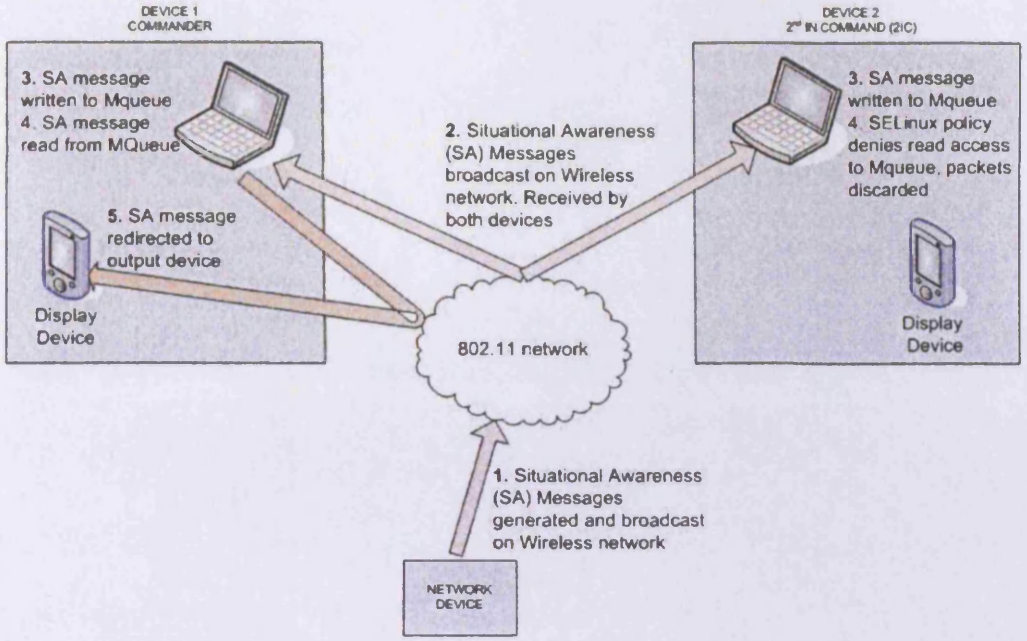


Figure 63: Flexible Security Demo when Device 1 is active

Figure 64 illustrates the system state from stage 4.

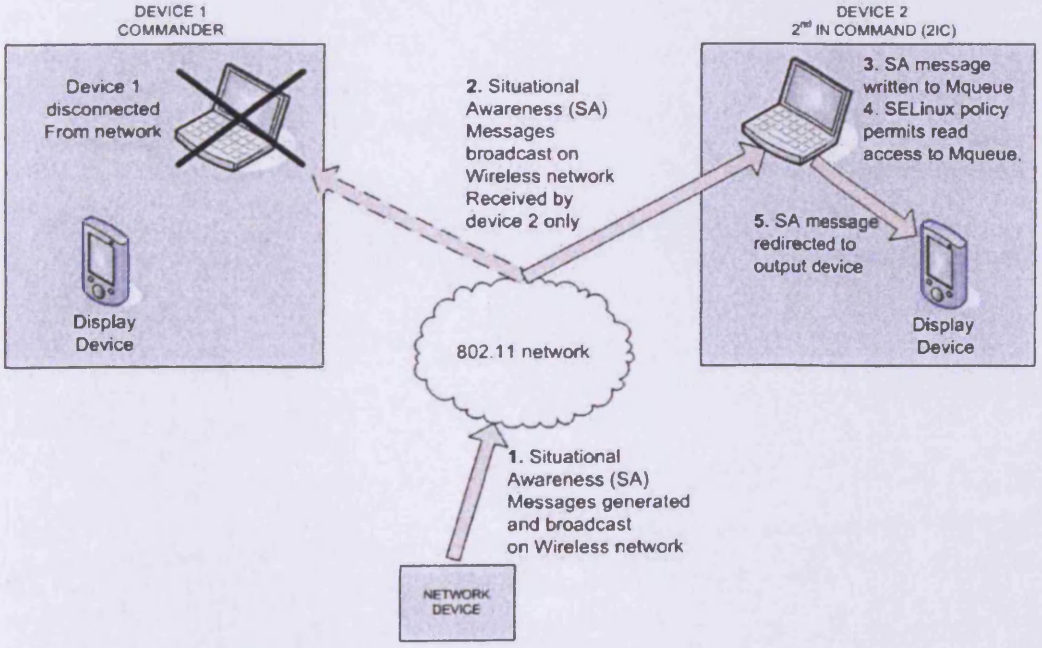


Figure 64: Flexible Security Demo when Device 1 is inactive

Although this new interpretation of the security requirements lends itself to a more visual demonstration, it does highlight an important tension between the theoretical models and the practical constraints of the target environment. We observe that

behaviours or characteristics arising from the implementation environment mean that the *actual* solution does not map exactly to the *original* requirement although the end result is the same. To clarify: all of the devices participating in our case study operate in a broadcast network, which is perfectly reasonable and usual for an ad-hoc environment. All of the messaging traffic is therefore received by all devices. In its purest sense, it could be argued that separation has not been achieved since Device 2, the 2IC, is also included in the network and will therefore be 'receiving' broadcast traffic along with its neighbours. However, Device 2 is not permitted to read the message queue therefore it has no knowledge about the type or content of the actual message. Strictly speaking sharing has been denied for the 2IC in this situation, but obviously at a higher layer in the communication hierarchy. Ultimately the debate comes down to how strictly the system designer wishes to interpret (and enforce) 'sharing' and 'separation'. Since our work is concerned with balancing flexibility with traditionally highly segregated solutions, separation must occur at a higher level than the physical layer in order to support expedient reconfiguration. The implementation of 'logical' separation over 'physical' separation presents a significant engineering challenge. This is particularly the case in environments such as the military where the separation between resources is of paramount importance to prevent contamination or compromise of sensitive information. In our solution, the properties of the physical environment necessitate the use of separation techniques at a higher level. This constraint is actually an opportunity to manipulate the underlying infrastructure, through security policy, and support our flexible solution. We only dwell on this point in order to underline the importance of being aware of characteristics in the implementation environment that will undoubtedly influence the engineering solution.

6.1.2.1 Environmental Constraints

In the previous section we described how the properties of the broadcast network affected the level at which separation will be engineered for our practical demonstration. Furthermore, the target system operates with limited bandwidth, power, storage and processing resources with the result that our solution must have a minimal footprint and tightly control the amount of additional overhead that is placed on the system. Integration of the NPC2 must therefore be innovative and incorporate

ways in which existing data and resources can be used to provide input to components such as the NPC2 Environment Monitor and Event Manager. The target system operates on a secure version of Linux (SELinux) and our solution leverages the security features of this operating system to define and construct the security policy components of the NPC2.

The system is designed to operate in a mobile, ad-hoc community but also has the capacity to communicate over fixed links. Our solution is focussed on the ad-hoc network where community members join and leave fairly regularly and for varying durations. Since our case study is concerned with using the absence of the 'Commander' (Device 1) to drive a policy change we must consider the impact of using a potentially highly mobile community of users and set our acceptance threshold for the event 'user absent' accordingly. If this is set too low then we risk changing the security policy unnecessarily.

In an ad-hoc network routing protocols including the Ad-hoc On Demand Distance Vector (AODV) protocol [66] are used to discover the existence of other group members in the network and support message routing. All community members can 'see' their immediate neighbour, that is, a device 'one hop' away. If the address of a remote member is known, a route request can be issued to locate another user. These characteristics offer both challenges and opportunities for our solution. It cannot be guaranteed that the device held by the basic user, who will assume the status of the agile group leader, will also be its nearest network neighbour; that is, one hop away. The device will only be aware of the existence of the leader if a network route has previously been established between the two devices. It will then, as a matter of course, receive route error messages when the agile leader device can no longer be reached. As part of the normal communications procedure it is expected that all devices would establish an initial communication with the leader device and therefore a network route would be set-up. This is probably an operation that should be automated to ensure that the required routes are in place. As part of the network configuration the basic user devices must have some means of resolving the network address of the agile group leader to identify the actual device and user. This is required to assist with observing and reporting the absence or presence of the leader and enabling the 2IC to assume elevated capabilities when necessary. For the first

demonstration we have elected to pre-configure some of this information so that, for example, the 2IC device has prior knowledge of the Internet Protocol (IP) address for the Commander device and can detect its presence in the network using routing information generated by AODV. In addition a pre-defined list containing the IP addresses of community members is read by the Environment Monitor to determine the identity of the device which will assume the capabilities of the Commander when the security policy is updated. Whilst it is recognised that this approach is contrary to the normal functioning of ad-hoc communities, it is also appreciated that expending effort solving complexities generated by the external environment would have jeopardised and detracted from development of the NPC2 itself. The military community does operate in a hierarchical manner therefore using a pre-defined list to determine the 'next in command' is representative of normal working practices.

6.1.2.2 Policy Management

In our ad-hoc network configuration there is no central point for administration or management and this clearly impacts how much local management will be necessary in the deployed solution. One of our objectives is to minimise the amount of intervention that is required by the human 'user' or administrator. It will therefore be necessary to have the components necessary to enable a change in behaviour installed or accessible locally but protected by suitable security mechanisms and only invoked when policy change is triggered. In support of our research objectives and philosophy this approach is acceptable. It is also completely in harmony with Ashby's Law of Requisite Variety [5] since we are striving to deploy a system with the capacity for different options and thus increase its lifespan before external intervention is required.

6.2 Implementation Components

The NPC2 is designed to interact with and add a flexible dimension to existing systems. It is therefore necessary to provide some context for the NPC2 and identify areas where the modules defined in the high level architecture overlap with the host system. Much of our discussion has focused on the NPC2 manipulating the underlying security protection mechanisms however we have not, as yet, described the resource that is actually being protected. Figure 65 illustrates the demonstration

environment for the NPC2. Here it can be seen that the system uses a message queue (MQueue) which will be protected by the security policy.

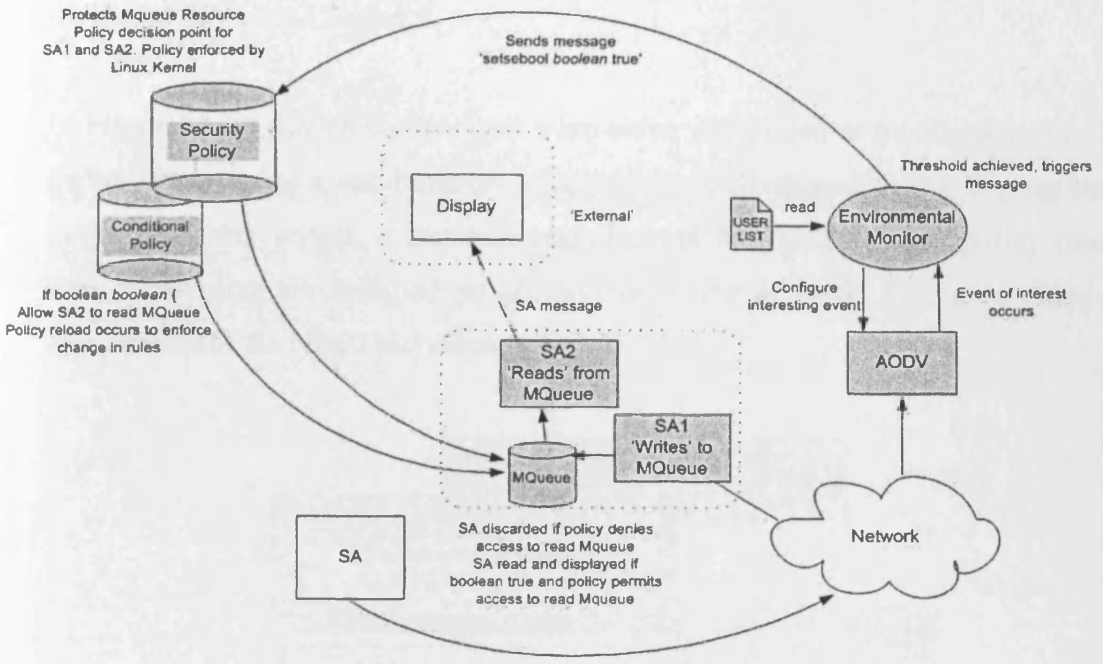


Figure 65: NPC2 Demonstration Environment

Protection of MQueue is achieved by controlling the access of two processes SA1 and SA2 which have write and read access respectively. Not only do MQueue, SA1 and SA2 exist in the external environment they are also regarded as part of the test environment. Without them, the functionality of the NPC2 cannot be verified. The development of MQueue and related components in the external environment is acknowledged throughout this chapter. However, a detailed description of the implementation is reserved until Section 6.2.5, which is devoted to the test environment.

Recall that Figure 60 illustrated a combination of broken and solid arrows between the Environment Monitor, Event Manager and Policy Manager components. This notation has been used to suggest that one or more data streams may be used as input/output to each process. For example, the Event Manager may receive a number of data observations from the Environment Monitor that it fuses and processes into one policy trigger. This could result in a single policy rule change or many, depending on the requirement. Our demonstration is concerned with a single event

and does therefore not require complex processing by any of the NPC2 components. However, the management of multiple events is fundamental to scalability in the NPC2 approach and should therefore be captured in the more detailed models. This is discussed further in Chapter 7.

In Figure 65 the shaded components were either developed or modified for the NPC2 demonstrator using a combination of applications developed in the C programming language, Linux scripts, a standard text file and SELinux security policy modules. The components are installed on all devices in the network. Figure 66 maps these components to the NPC2 architecture.

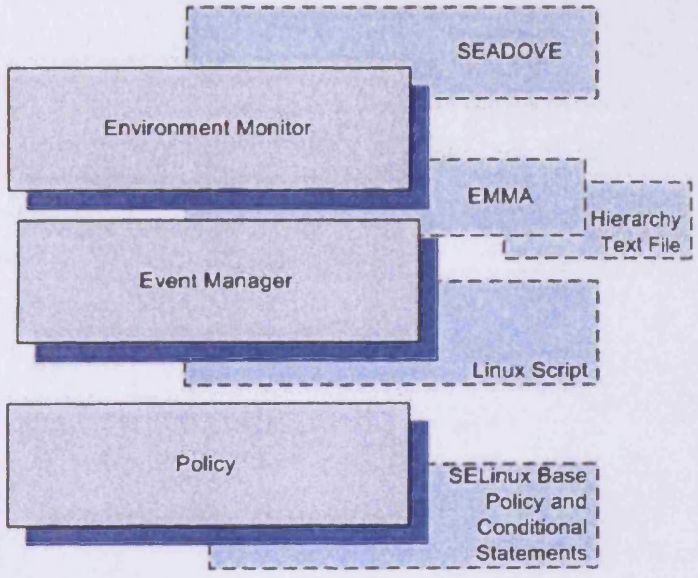


Figure 66: NPC2 Physical Components

In Figure 66 the shadow boxes represent each module identified in the NPC2 architecture. Behind each there is a box naming the software component developed for the demonstration system. This is with the exception of the external environment components SA1 and SA2. These are implemented as applications also developed in the C programming language and named SHIMMER and SHIMMEE respectively. These applications are described in Section 6.2.5. The development of the other NPC2 components is described in the sub-sections which follow.

6.2.1 Environment Monitor

As described in Chapter 5, the Environment Monitor (EM) is responsible for observing events of interest, collecting this data and feeding it into an analysis process. In our prototype NPC2 the EM is a software-only solution satisfied by two components: SEADOVE and EMMA.

SEADOVE

This is an enhanced version of the ADOV routing protocol designed for use in Ad-hoc networks. As stated in Section 6.1.2, for the purposes of our demonstration we are interested in events related to the Commander (device) and specifically when it is no longer available. In order to reduce the amount of overhead introduced by the NPC2 it was decided to utilise the routing and connection information provided by AODV and stored in routing tables as part of its normal operation. Having modified the standard protocol we referred to it as SEADOVE and to avoid confusion use this name throughout the chapter. The majority of the data processed by SEADOVE is related to network routing. It therefore includes AODV messages such as Route Request (RRQ), Route Reply (RRE), Route Error (RERR) and so on. These messages and their format are described in [66]. For the purposes of the NPC2 two functions were added to SEADOVE to perform housekeeping and call-back activities. The housekeeping function maintained a list of network subscribers by their IP address. This allowed the absence of the 'subscriber of interest' or the loss of this IP address from the subscription list to be detected. The call-back function in SEADOVE simply informed EMMA when the event of interest, that is the loss of the IP address associated with the Commander device, occurred.

SEADOVE is implemented as a Linux kernel module and EMMA a user-space application. A means of communication between the two processes is required. This is provided through the use of a Netlink socket created specifically for the purpose. Only the EMMA and SEADOVE processes use the socket which, given the requirement for secure messaging in the NPC2 architecture, is an important point. An attacker would need to be aware of the existence of the socket and once they had this information it would be possible to intercept messages passed between the two processes. Whilst it is unlikely that this would occur on the machine, it is nonetheless

possible. No claims can be made regarding the security of this regime and for a more robust solution a different approach would need to be found.

At system start-up a conversation occurs between SEADOVE and EMMA the purpose of which is to register an interest about a user of interest. To achieve this, EMMA parses a text file listing the IP addresses of network members and passes the details of those it is interested in to SEADOVE. The housekeeping function within SEADOVE maintains this list. As part of the subscription process, EMMA instructs SEADOVE to send a message when it detects an event of interest related to those members. SEADOVE observes and gathers routing information. Software code located within SEADOVE determines whether it is 'of interest' or not. The call-back function which is a software hook inserted into the AODV route management table, instructs SEADOVE to inform another device about an IP address of interest, in this case the IP Address associated with EMMA.

EMMA

EMMA performs a dual role in this implementation. Not only does this software component interface with SEADOVE to provide environment monitoring, it also runs code related to the Event Manager function. As described above, EMMA subscribes to SEADOVE in order to be informed about events of interest. In this case the event of interest is the absence of the Commander device as detected in the route management tables maintained by AODV. When EMMA receives the message from SEADOVE informing it that the event has occurred, the process consults an external text file to determine who will assume command. This file is essentially the hierarchy of members in the operation and contains the IP Address information of current network members. EMMA parses the list to determine who is 'next in command'. This effectively determines the device on which the security policy will be updated to allow the message queue read process to read and share situational awareness messages from the network. Having established the next in command and assuming the 'next in command' is the host device, EMMA runs a standard Linux script file which effectively changes the role from EM to Event Manager (EvM) and un-subscribes from the SEADOVE process.

6.2.2 Event Manager

The Event Manager (EvM) is a software solution satisfied by two components: specifically EMMA and a standard Linux script file.

In the capacity of EvM, EMMA calls the external standard script file. Given that the NPC2 demonstrator is hosted on SELinux for our example, it was necessary to interface with the conditional policy extensions which are a feature of the SELinux security framework. Conditional rules work with Boolean variables. The system administrator defines a Boolean variable that will be associated with the conditional rule set and sets the default value. These are activated when the value of the Boolean variable is changed. In our example, the script file run by EMMA contains a 'setsebool' statement which modifies the value of the Boolean 'userread' from its default setting of 'false' to 'true'. It is this change in value which causes the next part of the NPC2 to execute – the conditional policy statements in the SELinux security policy itself.

6.2.3 SELinux Base Policy and Conditional Statements

The NPC2 components are hosted on a platform which itself operates under the control of a security policy. As a result there were a number of factors to be considered as part of the security policy development. For example, the components of the NPC2 must be defined within the existing protection profile so the target platform literally permits the new applications to run and access the resources they require. Furthermore, the test environment which is manipulated by the NPC2 is hosted on the target platform and must also be covered by a suitable security policy. In short, the following tasks had to be considered and addressed as part of the security policy development phase.

- Host platform security
- Security of the test environment
- Application Security
- Creating the conditional environment

This is presented in Figure 67 which summarises the policy configuration required for the demonstration.

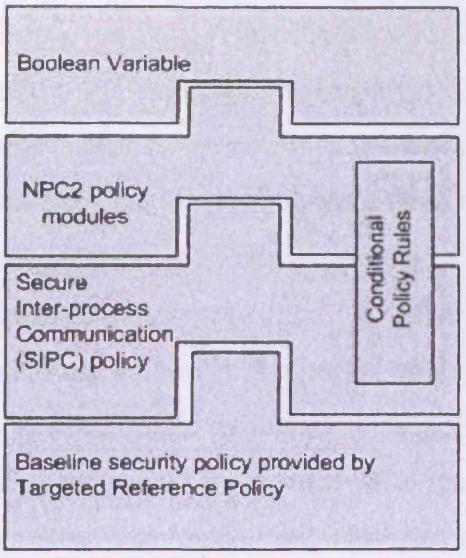


Figure 67: NPC2 Policy Structure

Although logically the NPC2 policy modules are considered as entities separate from the baseline security policy, in practice they are a part of the overall package enforced by the operating system kernel. The relationships between the conditional rules and the Boolean variable mean that the conditional rules within the policy are not active (unless the Boolean value is True) although the policy module itself, where the rules are defined, is both loaded and active. The second task, developing security policy for the test environment, is described in Section 6.2.5. The remaining three are discussed in the sections which follow.

6.2.3.1 Host Platform Security

The NPC2 demonstrator was hosted on the Fedora Core 6 version of Linux with SELinux enabled. After installation, the SELinux reference policy was downloaded from Tresys [67] and used to build the baseline security policy. As described in the Literature Review, a significant amount of effort has been invested by third-party developers in creating and maintaining a Reference Policy for SELinux installations. This is a set of individual policy modules designed to secure applications and services that could reasonably be expected to run on a Linux system. To assist with policy

development, it is possible to select either the *targeted* or *strict* policy builds and then customise in accordance with the organisation's security goals and the requirements of the target platform. For our demonstration, the *targeted* policy was selected. This contains policy modules to secure network related services and applications. SELinux can be run in one of two modes: *enforcing* and *permissive*. During the development of security policies *permissive* mode is generally used because this records policy violations in the audit logs but does not stop the system from running. The developer can utilise audit messages to update the policies accordingly and rectify bugs. Once the system is ready for deployment *enforcing* mode is generally switched on. This mode also records policy violations; however the kernel enforces the rules as defined. Quite clearly, if mistakes have been made in the security policy modules the system can quickly become unusable, even to the Root user! The results recorded in Section 6.2.6 were collected once the enforcing mode had been turned on. If the device had been operating in permissive mode the access denied message shown later in Figure 71 would not have been displayed (because the policy would have allowed this activity whilst recording the violation). This message was generated when SHIMMEE attempted to read the message queue on the 2IC computer.

SELinux is a good development environment for the NPC2 security policy because it provides flexibility and allows an administrator to tailor the security configuration in accordance with specific requirements. New policy modules can be created as organisational requirements demand and loaded either as part of the baseline build or independently during run-time. However, there are benefits over and above these practical issues. In the literature review we described how SELinux enables flexible Mandatory Access Control through type enforcement. Every object on the system is assigned a 'type' as part of its security context (which in full comprises *user id: role: type: security classification*). The *type* is the most important determinant in permitting access. For example, if a subject's *type* is not authorised for an object's *type* it will not be permitted access irrespective of the user id or their role on the system. This is precisely why the Root user can effectively lose access on the system when enforcing mode is switched on, if the security policy has not been defined correctly. In each of the new security policy modules created specifically for the NPC2 we declare the 'types' necessary to allow each application to run and declare the types for any system resources it requires to fulfil its task. Types are associated

with physical objects and different methods of object access such as read, create or write are specified. Consider the following extract in Table 2 from the security policy module for the EM 'EMMA'. For ease of reading line numbers have been inserted.

```
1    ####################
2    # Declarations
3    #
4     type emma_t;
5    domain_type(emma_t)
6
7     type emma_rw_t;
8     files_type(emma_rw_t)
9
10   ####################
11   # local policy
12   #
13   Allow emma_t emma_rw_t : file manage_file_perms;
```

Table 2: Example of SELinux policy for EMMA application

On lines 4 and 7 the domain types emma_t and emma_rw_t are declared. These are associated with physical objects on the system through a separate but associated *file context* policy file.

Line 13 is an access control rule which specifies that any object with the source type of emma_t is granted the permissions specified in the macro manage_file_perms to a class type 'file' of target type emma_rw_t.

In this example, if there were other objects assigned the emma_t type they would also acquire these permissions to any object labelled with the emma_rw_t type. As it happens, our policy is restricted, only the Environment Monitor EMMA and the associated script file have these labels therefore no other object has access. For example if SHIMMER attempted to write to the script file it would be denied access. SHIMMER has a different type mq_sender_t which is not authorised for the emma_rw_t object type.

Under this regime it is possible to separate objects on the system in accordance with the type they have been assigned. Effectively the 'type' defines a virtual boundary or container to hold all objects with the same designation. The NPC2 components are

isolated from each other, unless access has been specifically enabled through an external interface, and from other objects on the system.

6.2.3.2 Application Security

As stated previously, it was necessary to create security policy modules for each of the NPC2 applications to allow them to run on the target platform and access the necessary resources to fulfil their individual tasks. The new modules were compiled as part of the SELinux targeted reference policy build, and the overall policy itself was built as a series of loadable modules. For the purposes of development this is a useful approach since it allows us to load and unload modules independently of each other and make changes as necessary. Once the system is deployed, the developer can select between the loadable module approach or a single monolithic policy file. There are advantages and disadvantages with each approach and the choice would very much depend on the nature of the deployment, the extent to which policy maintenance was expected and the expertise available to manage the system.

6.2.3.3 Creating the Conditional Environment

As stated in [57, p.186] "...the ability to change Boolean variable values in a running system is what enables us to vary the value of conditional expressions and hence gives us conditional policies...". It is precisely this functionality that we leveraged as part of the NPC2 installation. In our autonomous solution the act of defining Boolean variables and adding conditional statements to the security policy module was only one part of the task. The value of the Boolean variable needed to be updated 'remotely', that is, by some means other than keyboard input by a human user. The SELinux language uses the command 'setsebool' to change the value of Boolean variable and commit the change on the system. This action causes the operating system kernel to use the new value immediately without a system reboot. In our solution, the setsebool command and associated parameters were put into a script file. This file is called by EMMA whilst acting in the role of EvM.

The Boolean variable needs to be defined and assigned an initial default value. This was achieved by manually inserting a new Boolean 'userread' in the Booleans file

which also has the advantage that the Boolean will persist across system reboots. The default value was set to false. The script file called by EMMA contains the following statement which clearly changes the Boolean value to true.

```
setsebool userread true
```

Since EMMA is running the script file and acting on behalf of the user, it is necessary to provide the process with the appropriate permissions to carry out the task. The necessary rules were included in the security policy module to allow EMMA to run setsebool, write to the Booleans file and change the value of userread. Within the security policy module securing the message queue read process, a conditional statement was inserted as captured in Table 3:

```
if (userread) {
        allow mq_reader_t tmpfs_t : file read;
}
```

Table 3: Conditional Statement in SHIMMEE security policy

In the SELinux policy, the domain type mq_reader_t has been associated with the SHIMMEE application. Since the default value of userread if false, mq_reader_t and therefore SHIMMEE, will not normally be allowed to read the temporary file system where the message queue is written. Once the Boolean value is changed to true, this condition will be reversed. Figure 71 in Section 6.2.6 shows the messages generated when this policy module is enforced.

6.2.4 Validation

Validation is a core feature of this research since we are seeking to provide a balance between the pursuit of flexibility and the requirement for assurance. In earlier sections we described how the NPC2 itself could be used in the risk assessment process as a means of revealing any weakness arising from modifying the separation and sharing profile of a system. Note that for the purposes of our research, the NPC2 has not been subjected to the rigorous software engineering approach that would be expected and required from a fielded system or for security certification. We therefore

explore different aspects of the design that lend themselves to validation and identify where specific measures have been put in place to assist with this process. Validation has been considered at a number of levels, specifically:

- Logging and auditing
- Validation of security policy
- Testing

6.2.4.1 Logging and Auditing

The Environment Monitor and Event Manager components, which were developed specifically for the NPC2, produce output which can be directed to the screen, an application log, or both. Error messages and application logging messages are both captured, and provide some confirmation of the application's behaviour as it is running. The ability for each application to be able to write to its own log is an activity that must also be included in the SELinux security policy. Unauthorised access attempts are recorded in the SELinux audit logs. This allows the system administrator to monitor and confirm that only the NPC2 components themselves are able to use the logs. Note that SELinux access denial messages are recorded in the logs, that is, when the conditional statement is activated to allow SHIMMEE to read the message queue we do not see a corresponding 'access granted' message. However, events related to the change in Boolean value and subsequent policy reload are both recorded. An extract from the audit log file taken during one of the test runs is shown in Table 4.

```
type=MAC_CONFIG_CHANGE msg=audit(1194444205.117:41): bool=userread val=1
old_val=0 auid=0
type=USER_AVC msg=audit(1194444205.147:42): user pid=3320 uid=0 auid=0
subj=root:system_r:unconfined_t:s0-s0:c0.c1023 msg='avc:  received policyload notice
(seqno=3) : exe="/bin/dbus-daemon" (sauid=0, hostname=?, addr=?, terminal=?)'
type=SYSCALL msg=audit(1194444205.117:41): arch=40000003 syscall=4 success=yes
exit=2 a0=6 a1=bf8bee36 a2=2 a3=bf8bee38 items=0 ppid=3543 pid=3544 auid=0 uid=0
gid=0 euid=0 suid=0 fsuid=
```

Table 4: Extract from Audit Log file

Figure 68 is a screen shot of the access denial message recorded in the audit log when SHIMMEE is not permitted to read the message queue. To obtain Figure 68 the seaudit tool developed by Tresys [68] was used. This provides a graphical interface and accessible means of viewing SELinux messages written to the standard Linux audit log file.

Figure 68: Screen shot of AVC error using seaudit graphical tool

6.2.4.2 Validation of Security Policy

Within the SELinux community there has been a good deal of activity to develop tools which assist in the development and validation of security policy. The SELinux Integrated Development Environment (SLIDE) is one example to emerge from ongoing research and development from Tresys [69]. This is an open-source graphical tool running as a software plug-in to Eclipse [70] designed to assist the user with writing (or modifying) security policy modules. As part of the build process various validation routines are performed on the security policy code which check for faults such as syntax errors which will prevent the policy from compiling. Of course, successful compilation does not mean that the policy is good or even accurate therefore it is necessary to confirm that it does secure the right resources in the required manner. As described in Chapter 2, Linux is a complex operating system therefore the SELinux policy will itself be complex in order to secure the plethora of objects and services on the target platform. This poses a significant challenge to the policy developer who is required to search through the rules looking for possible

inconsistencies or simply trying to answer the question 'who has access to what and where did the access came from'. To assist with validation, Tresys have developed an additional graphical tool named apol [71] specifically for policy analysis. This can be downloaded as open source software as part of the SETools download [72]. It should be noted that the tool itself requires an investment in learning before real benefit can be derived. However, having mastered the interface it is possible to drill down into a policy and perform validation checks. Figures 68 and 69 provide examples of the policy analysis tool in use to show a couple of cases from our compiled security policy for the demonstration. The search criteria used to obtain Figure 68 verified the policy rules associated with the source type emma_t on the target type emma_rw_t. It can be seen from Figure 69 that a total of 2 rules match the search criteria. They also confirm the rules declared in the EMMA security policy module. Had the policy analysis tool returned more rules it would have been necessary to track back through the policy to determine the origin of what would be, in this case, an error.

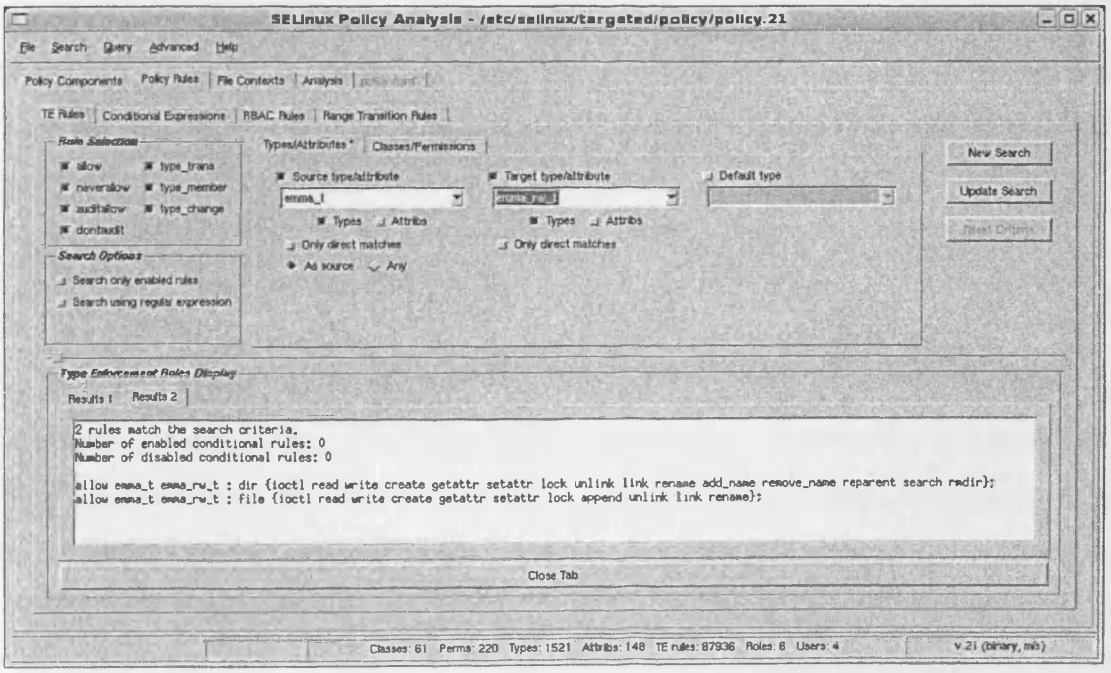


Figure 69: Screen shot from apol policy analysis tool

Figure 70 is another example taken from the policy analysis tool but in this case we are looking at the permissions of the type mq_reader_t on the temporary file tmpfs used by the message queue. Recall from the previous section that mq_reader_t is associated with the SHIMMEE application.

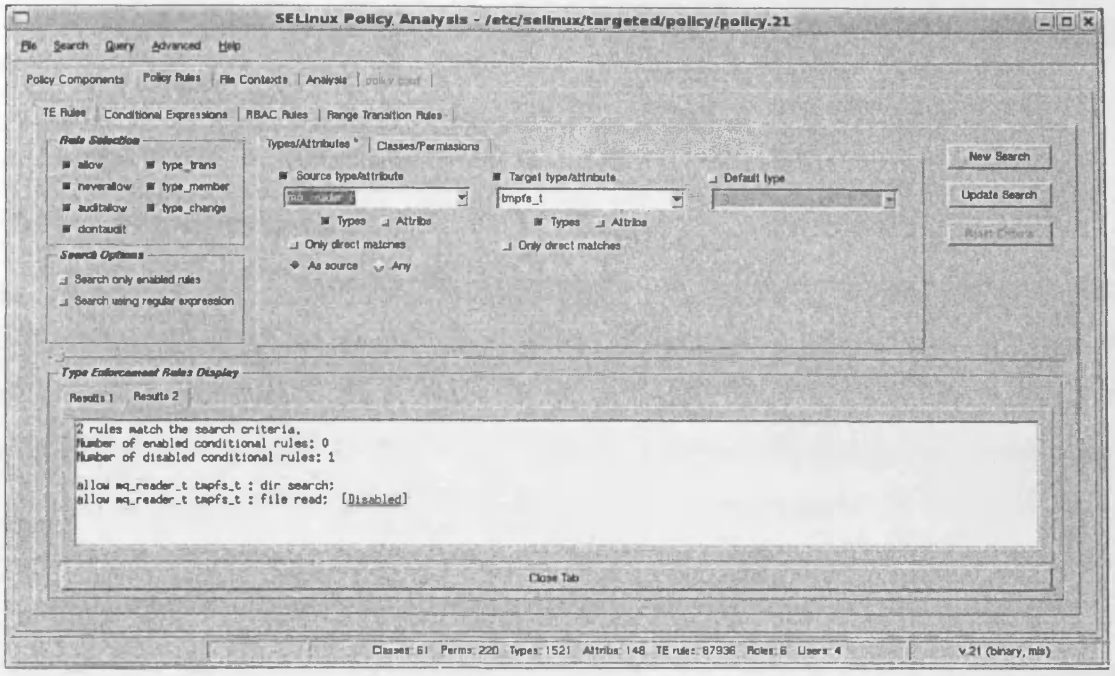


Figure 70: Screen shot from apol showing access control rules

Figure 70 shows quite clearly that the rule displayed is a conditional rule which is currently disabled. The results in Figure 70 confirm our expectation of the security policy for mq_reader_t; that is, it only requires read access to this object and any other result would have been erroneous.

6.2.4.3 Testing

For the purposes of our research it was necessary to perform tests to confirm the correct functioning of the NPC2 applications and correct application of the security policy in accordance with our case study. As described above, validation of the security policy also forms a part of this testing. It is recognised that for a more mature development of the concept and certainly if evaluation of the solution were being sought, more rigorous testing both of the components and the security policy would be required. For example we would expect to utilise the results of the risk assessment process, where different threats are identified, as a means of devising a range of penetration tests. This approach would enable us to confirm the strength and applicability of the security controls proposed during the risk assessment phase and harden the defences as appropriate. A full scale set of penetration tests would be inappropriate at this stage of the development; it is sufficient to conduct preliminary

tests to confirm the initial design and functionality of the NPC2. The test results are described in Section 6.2.6 which contains and discusses experimental results from one of the demonstrations of the NPC2.

6.2.5 Test Environment

In Figure 60 the NPC2 was shown in the context of the external environment. This diagram identifies protected resources that is, those objects on the system to which access will be controlled via the security policy. Of course it is this protection that will ultimately be manipulated by the NPC2 as conditions change in the environment. In order to demonstrate the NPC2 in action a test environment is required which provides the resources and policies for the NPC2 to interact with. We recognise that the case study is contrived in that there was no test environment to begin with. In order to show the NPC2 it was necessary to create the environment to be protected in addition to the means for protecting it! As shown in Figure 61 temporary storage is required to hold situational awareness data received from other devices on the network. It was concluded the best way of achieving this was using a message queue. This also provided a physical resource which could be secured through security policy. Additional research revealed existing work conducted by Tresys on Secure Inter-Process Communication (SIPC) using SELinux [73]. The authors provided example applications and security policy modules to accompany the work which we subsequently used and adapted to form the basis of our test environment.

The SIPC package included management applications developed in the C programming language to create and destroy the message queue and applications to both write to and read from the queue. As shown in Figure 65, for the demonstration the policy decision and enforcement points are associated with the message queue read process. When this process makes a read request the security policy is consulted and the rule is enforced to either permit or deny access (depending on the status of the Boolean variable). If the access request is denied, a message is written to the audit log.

The applications from Tresys required little modification to run in our test environment. The most significant change was to the accompanying SELinux

security policy modules. It was necessary for example, to include permissions for services such as Netlink which was used in NPC2 test environment, but not the original. Our demonstration is based on a network configuration where data is being shared between devices. Every object on an SELinux system is labelled with a domain type including a representation of the network interface. The security policies were modified to include permissions related to networking services and protocols to enable the domain types to send and receive in this case, user datagram packets (UDP). The simple but most powerful change to the policy was adding the conditional statements to the policy associated with the message queue read process.

6.2.6 Experimental Results

The visual and dynamic nature of the NPC2 demonstrator causes some difficulties for presenting experimental results in a static medium such as this thesis or indeed conference papers. Whilst running the demonstrator it was possible to verify the correct operation of the NPC2 components because we could perform a visual check on the output devices. When the Commander Device was functioning in the network, the situational awareness data was displayed as a moving icon on the screen. At the same time, the icon was static on the 2IC device and recorded 'permission denied' messages when the read process, SHIMMEE, attempted to read the message queue. This message corresponds to an Access Vector Cache (AVC) denial message generated by SELinux and recorded in the audit log on the 2IC machine. We could therefore confirm that the security policy was being correctly enforced. When the Commander device was removed from the network, the NPC2 components functioned as designed. This was visually obvious because the situational awareness icon began to move on the 2IC output device once the policy change permitted the situational awareness messages to be read from the message queue. However, how can we illustrate the results of this dynamic behaviour in a meaningful way in the thesis? The answer is of course, that it is not possible. Therefore the evidence we provide for the demonstrator is contained in a series of messages taken from the 2IC device when the demonstration was run. These are captured in Figure 71 which has subsequently been annotated. An explanation of each number is provided beneath.

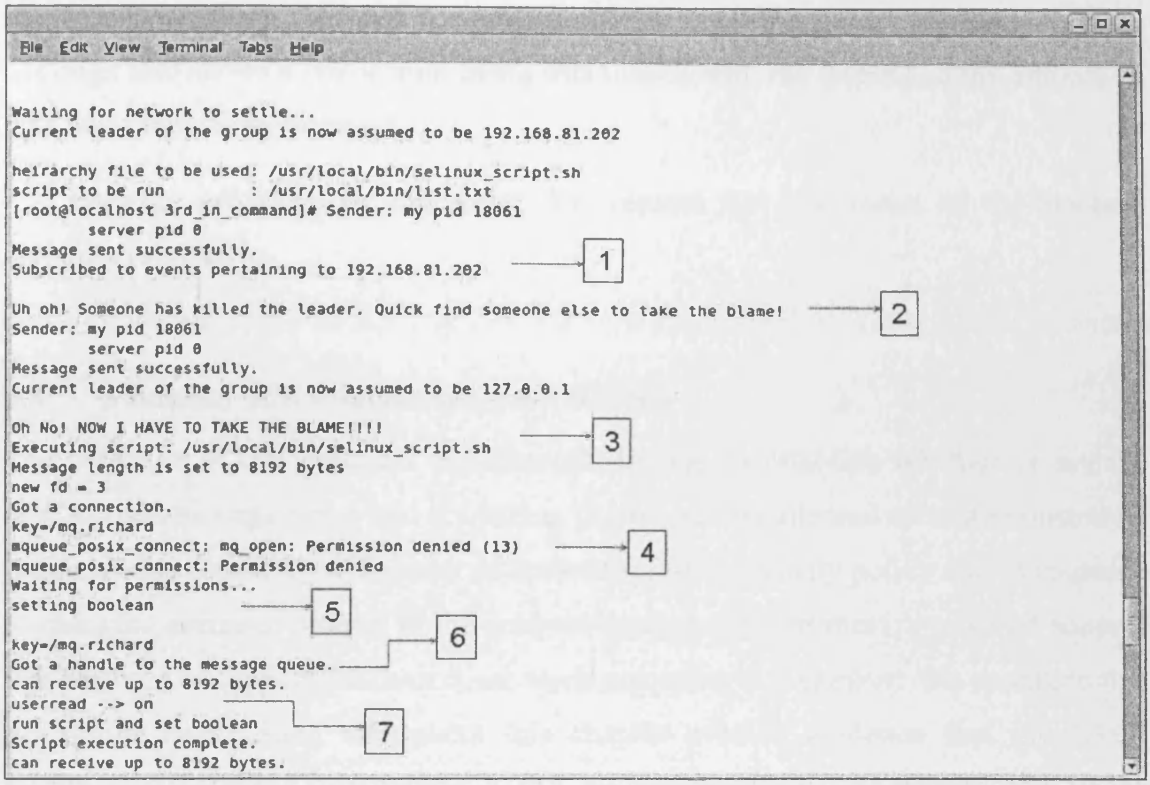


Figure 71: Screen shot from 2IC device; demonstration of NPC2 components

[1] message generated by EMMA when process subscribes to SEADOVE for events related to Commander Device.

[2] message generated by EMMA when process is informed that Commander Device is no longer available.

[3] message generated by EMMA acting in role of Event Manager. Running the script file will eventually trigger the change in security policy.

[4] message generated by security sub-system. At this point the SELinux policy denies access to the message queue for the read process and thus denies the ability to read the situational awareness data that would provide location (geographical position) information. At this point, the Situational Awareness icon will be stationary on the 2IC device.

[5] the value of the Boolean variable (userread) is being modified from false to true.

6 when 5 occurs the read process is able to read the queue and receive data. Message also shows a connection being established with the queue and the amount of data that can now be received.

7 message generated by the script file, returns the new status of the Boolean variable 'userread'.

6.3 Summary and Conclusions from Results

The objective of this practical experimentation was to establish whether or not the NPC2 could be engineered into a working prototype that allowed us to demonstrate a change in the capability of the user in accordance with security policy and in response to changing events of interest in the external environment. In short, a proof of concept was required to support the theoretical work preceding this chapter. We conclude that the results documented throughout this chapter provide evidence that the NPC2 architecture described in Chapter 5 can be implemented. Moreover, the whole concept of autonomously changing the capability of a user, in accordance with pre-defined conditions, can be achieved through our implementation of the NPC2 thus generating support for our original objective of engineering a more fit for purpose security solution. However, whilst this implementation has achieved what was originally intended, it is also recognised that there are a number of ways in which the work could and should be developed. The next section introduces some of the more important areas.

6.3.1 Enhancements to the Practical Experimentation

One area, which is an important aspect of the NPC2 architecture, is the security of messages between the various NPC2 components. Our initial prototype, which uses a Netlink socket to pass messages between SEADOVE and EMMA, does not include a security solution for the actual messages themselves. Since no other process uses the Netlink socket created for our purpose, it is unlikely that a message could be eavesdropped. However, we would not make any claims regarding the security of this approach and recognise that a more robust and secure solution needs to be found. This outcome does nonetheless underline one of the advantages and successes of the NPC2

architecture. Our method of implementation is only one of many possibilities. The NPC2 architecture is a pattern that can be applied to a problem and does not dictate any implementation methodology or technology. In another example the Environment Monitor and Event Manager could be constructed completely differently and therefore lend themselves to a more robust security solution for the messaging function.

As described in Section 6.1.2.1 for the first demonstration of the NPC2 it was considered appropriate to pre-configure some of the input data used by the NPC2 components. For example, the text file consulted by EMMA is populated with information pertaining to members of a community. Of course the whole philosophy of the ad-hoc network is that information is not known at the outset, and the devices discover each other and build up a picture of the community through the routing protocols. However, we judge that our decision to use pre-configured data does not detract from the success of the implementation for the following reasons. The primary objective for our prototype was to prove that the NPC2 architecture could be realised in a working solution and in order to do this, the various components require data with which to work. We considered that in the first instance, *how* the components acquired this data was a secondary issue to actually demonstrating that the concept of the NPC2 worked in practice. Furthermore, when constructing the demonstration it was necessary to pay attention to our target market (the military community) which operates within a strict hierarchy and pre-defined knowledge of other network members in their immediate group. Our use of pre-configured lists was therefore a realistic and reasonable approach to take for an early demonstration model. An extension of the NPC2 would work more closely with the characteristics specific to an ad-hoc network. We anticipate that in order for the NPC2 to function in a less deterministic manner it will be necessary to re-engineer components such as the Environment Monitor module to include greater decision making capabilities and 'intelligence'. For example, if the event of interest remains the same, that is loss of the Commander, the Environment Monitor may need to establish exactly who the Commander is within the network before it can setup the monitoring part of the process. If all devices are participating in the monitoring process they will of course need to share their individual results and corroborate that the Commander is actually absent and not just out of range to one or two users. The type of extension described would probably not be a realistic or appropriate solution for our target market at this

time, but it would be a useful piece of future work to allow the NPC2 to operate in other environments such as civilian emergency services for example.

At the presentation of [24] a question was raised regarding the revocation of rights, that is, what happens in the event that the Commander returns to the agile group? This scenario was considered as part of the demonstration but owing to time and resource constraints, not practically addressed. Theoretically it should not be an issue although it would need to be tested as part of further development. The NPC2 Environment Monitor registers an interest for certain users/devices therefore it can be configured to detect not only the absence of the Commander but also their return. However, this behaviour could introduce other challenges related to trust depending on how and why the Commander was absent initially. For example, the device and user may have parted company and its return to the network may be suspicious. It is expected that some form of authentication between the user and device and the device with other network users will be in place. This process would need to be successfully managed and communicated *before* the NPC2 policy trigger mechanism could activate a change in rules. Managing authentication is beyond the scope of this thesis. Nevertheless, this could affect the security policy and level of access permitted and without doubt, offers a good deal of scope for further work.

As part of its normal operation, the NPC2 would need to revoke previously granted permissions on the 2IC device and verify the success of this update. Given that the NPC2 architecture is technology independent it is assumed that the security policy can invoke any change in permissions that are necessary. In practice, the success of this operation is dependent on the functionality of the underlying security infrastructure which in our demonstrator is provided by SELinux. One component in the SELinux architecture is the Access Vector Cache (AVC). The AVC improves performance by storing access control decisions made by the security server for subsequent use. In theory it is possible to revoke previously granted permissions because the AVC is invalidated whenever a policy is loaded. Revocation does not work in all cases; for example access to connection-orientated sockets is only validated on initial access and not subsequent use therefore a change in policy will not be observed (Mayer *et al*, 2007, p.43). In our example the message queue 'read' call is continuous and constantly checked against the AVC. When the policy is reloaded via the NPC2

trigger mechanism the AVC is cleared. The new access rules are applied when the read process subsequently requests access to the message queue. When the Commander returns the replacement policy could essentially reflect the original rules, that is, access to the message queue is not permitted on the 2IC device. Alternatively, a completely different set of permissions could be enforced.

Without doubt the development of SELinux policies is a challenging task which requires substantial knowledge of the target platform, the application to be secured and the security policy language itself. This issue has been recognised within the community over the past few years. As mentioned previously, the *targeted* and *strict* SELinux policies were developed specifically to secure standard software components and services included with a Linux distribution. However, if an organisation has bespoke software which needs to be secured it is necessary to write new modules to cover these applications and services – not a trivial task. Furthermore the policy language and environment itself continues to evolve as enhancements are added to the supporting infrastructure or indeed as new policies are written. During the course of this research we encountered a number of changes which certainly hindered the learning curve in an already challenging environment. It should also be remembered that Linux itself, being an open-source solution, evolves at a fairly rapid rate. During the time of our investigation we started writing policies for the Fedora Core 4 distribution and presented our prototype solution on Fedora Core 6. This was by no means the most up to date version of the operating system at the time, either. Eventually we considered it necessary to establish a stable build on which to develop the policies rather than keep updating the environment. The security features of SELinux are integrated with the operating system kernel therefore changes to the build impact security. The decision to adopt software updates or patches is of course a problem that each organisation will address differently according to its own requirements and goals. It is not an area which should be ignored or taken lightly.

The SELinux community have been swift to develop toolsets and frameworks designed to ease the burden of policy management. There are graphical development environments such as SLIDE [69] which guide the policy writer through the process of developing a new policy module. The tool assists with naming domains for processes and resources required by the application. Furthermore, numerous

interfaces have been written and incorporated into SLIDE. When used, these pieces of code expand into appropriate access control rules, specific to the interface, when the policy is compiled. The policy developer obviously needs to know which interfaces to select and Tresys have been conscientious in providing meaningful names; but having done so, the developer is relieved from writing or modifying policy code unless specific tailoring is required. On a cautionary note, whilst these toolsets are successful in disguising some of the development idiosyncrasies, we would argue that a substantial amount of knowledge is still required to write policy modules with a degree of certainty they will behave as expected and provide the level of security necessary. We conclude that policy development is likely to present the largest overhead in evolving the NPC2 and potentially one of the biggest obstacles to acceptance of the SELinux implementation.

Chapter 7: Scalability of the Approach

A holistic approach has been adopted in this research where the NPC2 has been applied at an abstract level in the modelling and risk analysis phases and at a practical level in the implementation of a prototype system. This approach strengthens the research in that it demonstrates the potential of the NPC2 in different development phases of the fit for purpose security solution. In this chapter we explore the extent to which the NPC2 can be scaled in all aspects of our approach.

The first part of the chapter focuses on the modelling and risk analysis approach and in particular we look at how the concept of the NPC2 can scale in models that are either larger (involving more domains), more complex in the number and type of conditional connections between domains or indeed both. Our work with scalability and modelling continues by looking at the use and impact of the NPD concept in a larger configuration. The chapter concludes by investigating issues surrounding the development and implementation of multiple NPC2 components in a system and the effect this may have on the NPC2 architecture.

Scalability is an important issue for a number of reasons. One of the characteristics of the Information Age is the use of pervasive computing devices and the ability for a user to potentially construct and participate in numerous ad-hoc relationships for the purposes of information sharing: a trend which is beginning to blur the perceived size and boundaries of a traditional network. Without doubt our work with fit for purpose security needs to take account of scalability in terms of the relative size and complexity within the network and the types of implementation that it may be required to operate in. Moreover, it is necessary to look at the different domains of interest a user may potentially be involved with, particularly from the perspective of temporary membership, and assess the extent to which our proposed additions to the modelling techniques accommodate this behaviour. The NPC2 begins to capture the essence of the ad-hoc network by showing the existence of a *temporary* connection between different domains of interest and the NPD concept adds further definition through its representation of a temporary community that may be occupied by one or many individuals. Through the environment and portal constructs in the DBSy

notation we can show whether a domain, temporary or otherwise, can be accessed from the same platform or one which is separate through either technological and possibly geographical means. Given the increasing profile of ad-hoc networking it is important to explore the applicability of the NPC2 and NPD concept to a greater extent in this environment. In this section on scalability it is intended to draw out the relevance of these constructs in modelling ad-hoc and changing relationships between domain members. We believe this can reveal useful information regarding the suitability of a domain approach in the boundaryless or de-perimeterised world [2].

In terms of implementation it is necessary to understand how we would approach the task of engineering either multiple NPC2 installations into the network or indeed scaling the capability of one NPC2. We need to qualify how the constructs may interact with one another since their conjunction might produce quite unexpected and undesirable results. Of course the models have a significant role to play in revealing potential connectivity between domains as a result of introducing temporary changes to the separation and sharing profile. The technologies available to engineer a solution may prevent us from working with multiple NPC2 installations: issues which are explored in Section 7.2.

7.1 Scaling the Non-Persistent Capability Concept in DBSy Models

From our research it is known that DBSy collects individuals with compatible characteristics and information requirements into logical groups called domains and our models have, so far, been based at this level of abstraction. In Chapters 3 and 4 the parent domain for the agile mission group was refined into two sub-domains. This enabled us to accommodate the different external connectivity capabilities of the sub-domains and calculate risk accordingly. Initially we were concerned that the domain approach would be too coarse to capture the plethora of connectivity relationships which we believe are characteristic of the Information Age. Since domains may be occupied by one or many members, effectively the 'node' level can be represented, if necessary, through a single domain. It is recognised that a model could quickly become unmanageable if many individual exceptions are included. They should therefore be minimised as far as possible. Difficulties may arise when refinements to the model are captured within the connectivity matrices. Our approach of modelling

each sub-domain or child-domain as an independent entity enables an accurate representation of the relationships. However, as a result, the matrices may become very large, perhaps running into hundreds of nodes, and therefore difficult for the user to manipulate. Whilst computational problems can largely be overcome by technological means, large matrices may be cumbersome for the system designer to visualise and interpret meaning. This modelling problem is not confined to scalability and domain refinement; it is conjectured that similar challenges will present themselves if the different connection types are included in the risk assessment process and matrix calculations. It is therefore an aspect of scalability which needs to be addressed.

Our exploration of scalability also includes the NPD. In our case study we recognised that our initial model implied that all members of Domain AX were permitted to communicate with Domain B when the NPC2 was switched in this direction. This was not, in fact, the case. The NPD used in conjunction with the NPC2 allowed us to show the relationship between all of the domains, that is, that Domain AX does not communicate directly with Domain B but actually uses an ad-hoc domain which in reality would be occupied by the 2IC when the NPC2 is switched in that direction. Of course the NPD could be scaled in numerous ways to represent additional ad-hoc networks (and their relationship with the existing system) or indeed additional connections and connection types between existing domains. In this section we begin to explore the affect of doing so on both the modelling process and risk assessment.

The DBSy risk analysis process considers the type of connection between two domains and this identification assists the system designer in selecting a suitable security control to help mitigate risk. Our initial risk analysis in Chapter 4 deliberately ignored the type of connection since it was sufficient to focus the calculations on the existence of a connection rather than its use. Having demonstrated the affect of the NPC2 on compromise path analysis in Chapter 4 it is considered appropriate to conduct a more detailed investigation where, like DBSy, we also model the *type* of connection over and above the simple existence of a path. This will allow us to assess the impact of the NPC2 and its ability to scale up to manage different types of connection. As described in the previous paragraph, it is important to

determine and address any difficulties that scalability within the models subsequently introduces to the matrix calculations.

To summarise, scalability in the modelling and risk analysis process is investigated in three different ways. An enhanced version of the original case study is used to examine the following permutations:

- Additional NPC2 constructs
- Additional NPD connections
- Permitted connection types

7.1.1 Additional Non-Persistent Capability Concept Constructs

In this section the original case study is extended for the case where a conditional connection is applied between Domains B, C and AY. Figure 72 shows the InfoSec Architecture model and shows quite clearly that AY is permitted to communicate with Domains C and B; Domain B is permitted to communicate with AY and C and Domain C is permitted to communicate with B and AY.

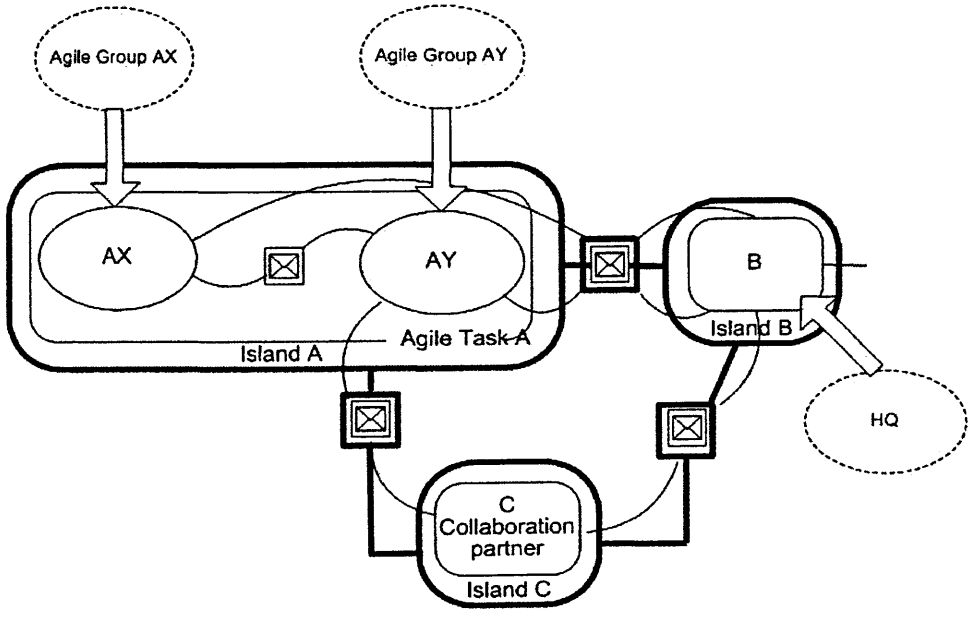


Figure 72: Model of extended scenario

A visual check of Figure 72 informs us that there are 10 potential compromise paths arising from the permitted business connections between domains. This assumes maximum permitted connectivity. However, the permitted connections in this scenario are conditional. The actual state of connectivity is captured in the revised InfoSec architecture model in Figure 73.

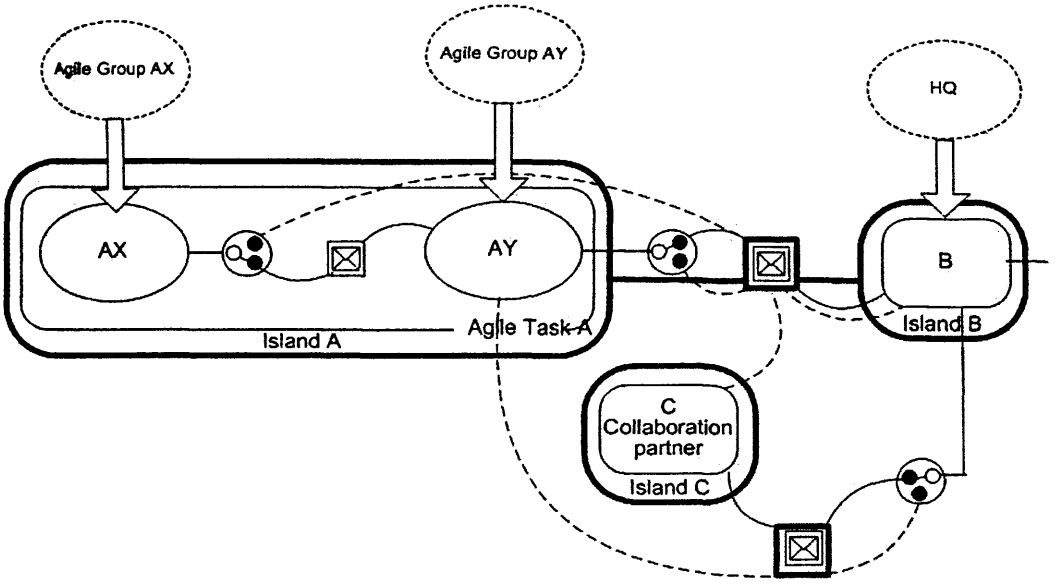


Figure 73: Revised case study model using NPC2

Figure 73 is based on the following requirements:

- Domain AX is permitted to communicate with either Domains AY or B but not both simultaneously. Figure 73 models the condition where the NPC2 construct permits connectivity between Domains AX and AY as shown by the position on the conditional connector.
- Domain AY is permitted to communicate with either Domains B or C but not both simultaneously. Figure 73 models the condition where the NPC2 construct permits connectivity between Domains AY and B as shown by the position on the conditional connector.
- Domain B is permitted to communicate with either Domains AY or C but not both simultaneously. Figure 73 models the condition where the NPC2 construct permits connectivity between Domains B and C as indicated by the position on the conditional connector.

The connectivity matrix C_{15} contains the direct, single step connections which could form potential compromise paths. As shown in (47) there are 6 potential paths.

$$C_{15} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \ldots (47)$$

(47) exhibits some interesting properties. For example, we have correctly modelled the requirements concerning simultaneous connectivity by allowing AY to connect to B but not C and allowing B to connect to C but not AY. However, this presents a problem when completing the connectivity matrix. It is shown that B has a connection with AY and C, which actually contravenes the policy. In fact this connection only exists because bi-directional connectivity is assumed therefore the path between B and AY is regarded as the 'return' path when communication has been initiated by AY – an interaction that is totally in accordance with the policy. This is a subtle but important distinction first identified in [23] because it illustrates the limitation with policy refinement from a DBSy model. Before requirements can be refined, it is necessary to capture all stakeholders and their potential interactions. In [23] we found that stakeholders can be identified from the DBSy models relatively easily but establishing potential interactions is more difficult since we can only see the existence or not of a connection. The example in Figure 72 is a perfect example of this issue since in order to understand the flow of communication and thus establish conformance to the security requirements, additional information about the interaction is required. To an extent it is possible to glean this detail from Figure 72 since the NPC2 controlling connectivity between Domains AY and B is clearly associated with Domain AY. Thus, when the connection is established an action will verify that a connection does not already exist with Domain C. However, this verification would not extend to confirming that Domain B has no ongoing connection with C before communication is established. It would be necessary for the policy conflict to be recognised at Domain B so before communicating with Domain AY, Domain B relinquishes its connection with C. Indeed this would be the case since the NPC2 associated with B would control the connectivity. Does this mean that the matrix

representation is incorrect or should we be content that the model is unable to capture the situation at the level required but fulfils a need in alerting the system designer at the policy refinement stage? This question is followed up later in the section.
(48) contains the results for the case C_{15}^2. This calculates indirect connectivity when the path length is equal to 2 and yields a total of 10 paths.

$$C_{15}^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \ldots (48)$$

Total connectivity for this configuration is calculated in (49) and gives 16 potential compromise paths.

$$C_{15tot} = C_{15} + C_{15}^2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \ldots (49)$$

When considering higher order connectivity some interesting points regarding potential compromise paths and the relationship to our defined security requirements arise. For example, in C_{15}^2 we can see that a connection exists between Domains AY and C but not B. Although this particular path was not defined in the model it is curious that it conforms to the security policy anyway! Furthermore Domain B has no connectivity with either AY or C and so its state has changed completely.

For completeness, we analyse the condition where the matrix contents accurately reflects the policy, that is, the NPC2 associated with Domain B prevents the return communication with AY because it, Domain B, has an existing connection with Domain C. In other words, there is no connection with AY even though AY is able to initiate the connection. C_{16} in (50) shows a total of 5 potential compromise paths.

$$C_{16} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \ldots (50)$$

This analysis moves our investigation into a different realm. Up until this point, all the connectivity matrices have been symmetric, that is the entries are symmetric about the diagonal. C_{16} is asymmetric and will therefore show different behaviour when the contents are enumerated in multi-hop configurations. In the case of 2 hop connectivity the following matrix C_{16}^2 in (51) is obtained.

$$C_{16}^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \ldots (51)$$

$$C_{16tot} = C_{16} + C_{16}^2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \ldots (52)$$

C_{16}^2 shows a total of 6 potential compromise paths. (52) calculates a total of 11 paths for this configuration which is significantly less than (49) where the symmetric version of the model revealed a total of 16 potential compromise paths.

Asymmetry occurs in the matrix when a communication between two domains is uni-directional. Until now, bi-directional communication has been assumed whereby two domains are able to communicate with each other albeit through a single point of control. The output in C_{16} simply indicates that the security requirements are being correctly interpreted because the NPC2 at Domain B is controlling simultaneous connectivity between Domains B, AY and C.

Uni-directional communication may be a result of deliberate planning, that is, a data diode is placed between two domains to limit information flow to one direction. An InfoSec Business model for this configuration is shown in Figure 74. It illustrates the business requirement where information may flow from Domain Low to Domain High-1 (as shown by the data diode on the permitted business connection) but not the reverse. Domains High-1 and High-2 have a two-way business connection. The matrix C_{17} for domains $\mathbf{D}_{17}$ = [Low, High-1, High-2] is shown in (53).

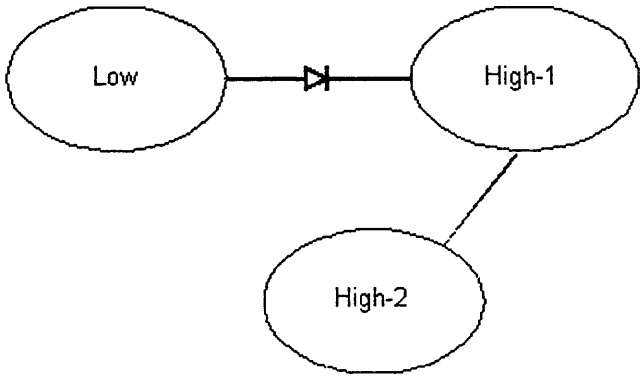


Figure 74: InfoSec Business model using diode

$$C_{17} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \ldots (53)$$

(54) calculates connectivity for the case C_{17}^2 where the path length is equal to 2.

$$C_{17}^2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \ldots (54)$$

Total connectivity is given in (55) and shows a total of 6 potential compromise paths.

$$C_{17tot} = C_{17} + C_{17}^2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \ldots (55)$$

This case is interesting because the higher order connectivity clearly shows the potential for a compromise path between the High and Low domains and from the InfoSec model it is known that the permitted business connection is in one direction only. This serves as a reminder concerning the trustworthiness of the one-way diode between Domains Low and High-1 – there is always the potential to exploit this means of communication if it exists. Furthermore, the case underlines the dilemma facing security architects that the benefit and requirements of sharing must always be balanced against the risk of compromise – albeit against seemingly considerable odds.

Having started a discussion on asymmetry in the connectivity matrices, the idea is extended to include an NPC2 and provide some conditions on the ability to share. It is quite conceivable for example, that the High-1 domain may, under certain circumstances, require the permitted business connection with the Low Domain to be bi-directional. However, we could further state that in this situation, it is no longer permitted to communicate with Domain High-2. The InfoSec business model for this configuration is shown in Figure 75.

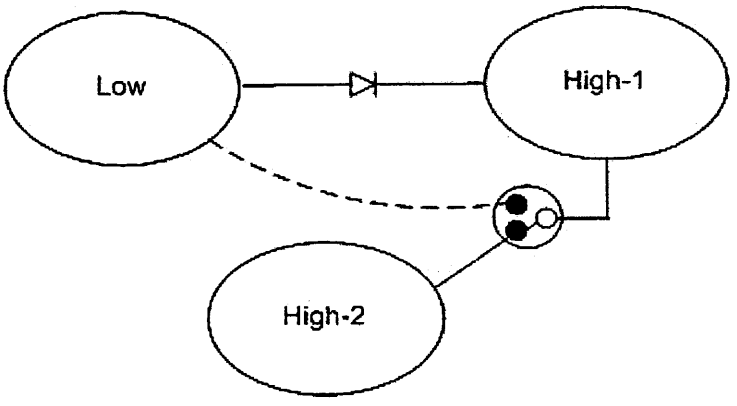


Figure 75: InfoSec Business model with diode and NPC2

There are of course some issues with this model. The NPC2 has been placed, quite correctly, adjacent to and under the control of Domain High-1. It shows quite clearly that normally it is permitted a bi-directional business connection with Domain High-2 as indicated by the position of the conditional connector on the NPC2. Exceptionally this would be changed to permit the temporary business connection with Domain Low at which time the connection with Domain High-2 would no longer be available.

However, the one-way connection between Domains Low and High-1 still exists which cannot be incorporated in this model. It is possible to modify the connections shown out of the NPC2 to connect both Domains Low and High-2 on the default path with Domain High-1. This does not imply that Domain Low is connected to and can therefore communicate with Domain High-2 through the conditional connector but that both domains are connected to High-1 when the conditional connector is switched in this direction. When the exceptional policy condition is activated, the conditional connector is switched to permit only the temporary path between Domains Low and High-1. This alternative configuration is shown in Figure 76.

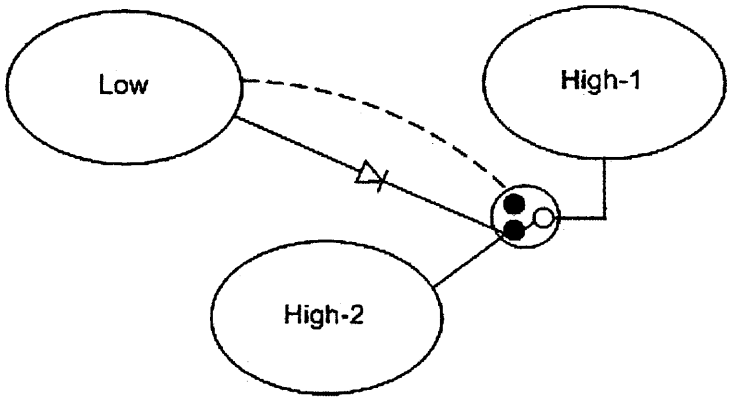


Figure 76: Alternate permitted path using diode and NPC2

In Section 7.1.3 a similar situation is described where the conditional connector is required to show two different permitted business connections. However, unlike the earlier models, Figure 76 captures a default case between two different domains. It is appreciated that the structure could become cumbersome and difficult to read if the number of connected domains continues to grow, and in such situations an alternative representation of the NPC2 may be required. A concluding observation arising from Figure 76 is the potential for error which may exist when interpreting the permitted connections. The NPC2 is associated with Domain High-1 as shown by the input point between the Domain and the NPC2. The NPC2 therefore controls connectivity between Domain High-1 and other domains connected to the NPC2 on the output point. However, it is recognised that there is a possibility of mistaking the connection between the output point on the NPC2 and Domains Low and High-2 as a permitted business connection which is not permitted in this case. It is proposed that familiarity

and training with the NPC2 notation will help to overcome potential misunderstanding.

The connectivity matrix for the exception to the policy is shown in C_{18} for domains $\mathbf{D}_{18}$ = [Low, High-1, High-2]. The Low and High-1 Domains have bi-directional communication and High-1 is not permitted to share with High-2. (56) shows 2 direct compromise paths.

$$C_{18} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \ldots (56)$$

Indirect, 2-hop connectivity for C_{18}^2 is shown in (57) and gives a total of 2 potential compromise paths.

$$C_{18}^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \ldots (57)$$

Total connectivity is calculated in (58) and gives 4 potential compromise paths.

$$C_{18tot} = C_{18} + C_{18}^2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \ldots (58)$$

These results indicate that the NPC2 has made a positive contribution to capturing the different conditions in Figures 75 and 76. It allows us to differentiate between the cases where a domain may need to swap from its default policy of uni-directional connectivity to a bi-directional configuration. Furthermore, the impact of this change on the number of potential compromise paths can be assessed through the matrices.

7.1.2 Additional Non-Persistent Domain Connections

This section broadens the investigation to explore the use of additional NPD constructs within the model and the case where additional connections between existing domains and the NPD exist. The aim is to qualify the extent to which this concept can be applied to ad-hoc networks where there is likely to be a proliferation of different and temporary communities set up for the purposes of resource sharing.

Of course one of the challenges of fit for purpose security lies in making predictions and assumptions about what may reasonably happen and how the system may need to change in order to accommodate changes to the separation and sharing requirements. There will always be an element of unknown; therefore it is believed that postulating about change and modelling the likely impact is a useful way of managing uncertainty and potential change in a controlled manner.

The next model presents the case where Domain AY has the capability to operate in an ad-hoc network where in reality this 'capability' may include community administration (establish, destroy) and/or community participation (exchanging information). At this level of detail we are only concerned that the domain may share information with the community using messaging when the capability is invoked. Note that in this variation of the model, Domain AY does not have a permitted business connection with Domain C except using the ad-hoc domain. However, it has a permanent connection with Domain B and a temporary indirect connection via the NPD. This configuration could be used as a means to temporarily share information necessary for all three parties but not information that Domain C, for example, would normally be permitted to see. The example assumes that other participants may include Domains B and C. These are actually outside of our scope of control, since the focus of interest is the agile mission group consisting of Domains AX and AY. Nonetheless, it is useful to reason about the potential risk if either one or both of these domains were connected. The temporary business connector has been used to connect Domains AY, B and C to the NPD construct since it would only be formed under particular conditions and probably as an exception to the baseline operation of the system. It does not have any dependencies with existing domains; that is, the policy does not state that AY may not hold a connection with the NPD and Domain B

simultaneously therefore it is unnecessary to include the NPC2 notation in this part of the model. Where used, the NPC2 permits Domain AX to share with Domain AY, which is the baseline policy. This arrangement is captured in Figure 77 where the domains are represented as $\mathbf{D}_{19}$ = [AX, AY, ADH_1, B, C].

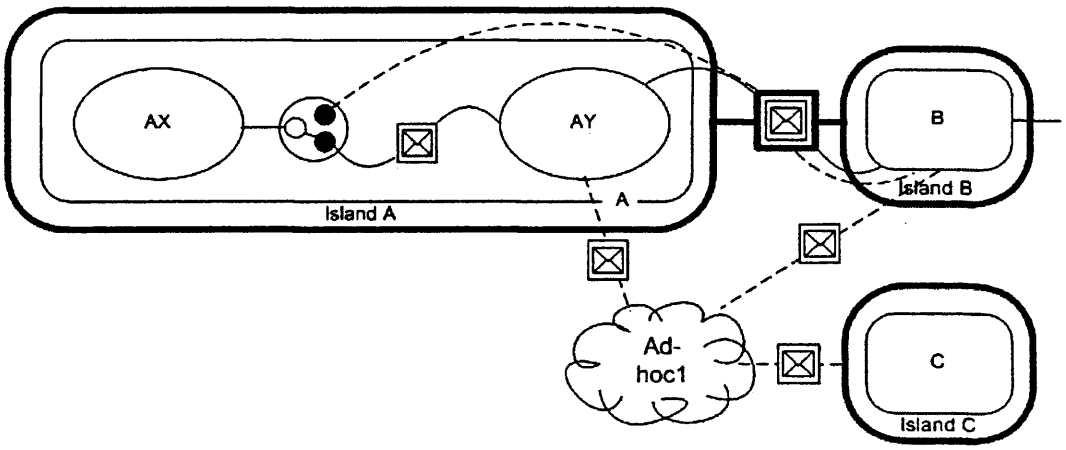


Figure 77: Multiple connections with NPD

The compromise path analysis of Figure 77 calculates both cases where the NPD is active or absent since this will affect the number of potential paths and the way in which security controls are applied. The results for matrix C_{19} are shown in (59) where the NPD (ADH_1) is active.

$$C_{19} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \ldots (59)$$

In (59) there are a total of 11 direct potential compromise paths. Connectivity for a path length of 2 is calculated in (60) and shows 28 paths.

$$C_{19}^{2} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 & 1 \\ 1 & 1 & 3 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad \ldots (60)$$

Total connectivity is 39 paths as calculated in (61).

$$C_{19tot} = C_{17} + C_{17}^{2} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 3 & 2 & 2 \\ 1 & 2 & 3 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \ldots (61)$$

In the complement of this case the NPD is inactive. Matrix C_{20} in (62) shows a total of 4 direct potential compromise paths. In this configuration, Domain C is not permitted to share with any other domain.

$$C_{20} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \ldots (62)$$

Finally (63) calculates C_{20}^{2}. The total number of compromise paths only increases by 3 from 4 to 7.

$$C_{20}^{2} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \ldots (63)$$

In (64) this gives a total of 11 paths.

$$C_{20tot} = C_{20} + C_{20}^2 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 2 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \ldots (64)$$

Of course there are other combinations of connections that could be considered. For example, ADH_1 could be active and in use by Domains B and C but not AY. Although this would not open a direct compromise path with Domain AY it would affect it indirectly via Domain ADH_1's connection with Domain B, or if we consider a hop count of three, the connection with Domain C.

From the work documented in this section, it is concluded that the NPD construct provides a useful method by which system stakeholders may speculate about possible connectivity between domains. The enhanced level of information can be used to make more informed decisions about the design of the security solution. The NPD can be used with multiple connections to existing domains in the model to assess the impact of enabling this capability on the domains in our focus of interest. Given the results above, the security analyst may decide that whilst Domain AY is permitted to work in an ad-hoc community this ability is subject to certain restrictions and they must relinquish the permitted connection with Domain B at this time. Importantly, with our approach, they can be armed with the information necessary to make this type of decision prior to the requirement arising.

7.1.3 Permitted Connection Types

In this section the investigation is extended to consider the type of connection which exists between domains. DBSy differentiates between email, shared storage and other methods of sharing data for business use. Extending our approach to consider the distinction between connections is an important aspect of the work since each will have a unique means of addressing risk introduced through the connection type in addition to sharing common security controls. Where multiple connection types exist, the number of potential compromise paths will also increase. The NPC2 is, by definition, a non-persistent capability. This alludes to an enforcement mechanism,

controlled by a policy, which permits a subject to perform a particular task. This could include establishing a connection for the purposes of information sharing, reading messages, writing to a data store or other activities. Granularity of capability is not provided in the DBSy models; instead the means of sharing data between domains is identified without specifying the interactions that define 'sharing'. This subject is addressed in [23] and [24].

This part of the analysis follows through the modelling process and applies our principles of expressing non-persistent connections using the NPC2. It is hypothesised that by combining the dynamic properties of a connection and its type we will achieve a greater granularity of expression with the DBSy models that will ultimately be reflected in the policy itself. Furthermore, this more detailed analysis will influence the technical opportunities for introducing more 'intelligence' into the NPC2 when it is realised in a tangible form. For example, a connection between two domains could be modelled where sharing is permitted using email but not database unless conditions in the environment dictate otherwise. As shown in this section, the type of situation described can be captured using the NPC2. Although this connectivity can be captured in a matrix (or possibly sub-matrices within a single matrix) as part of the risk assessment process, implementation of such capability requires the ability to detect the need for two methods of sharing and resolve any conflict that may arise. The solution may require more than one NPC2 in a practical solution with an implicit requirement for communication and co-operation between the installations. The examples in this section allow us to explore and comment upon any ramifications this may pose for the notation and practical implementation.

Figure 78 includes different types of connections between domains. The original scenario only permitted messaging capabilities. In Figure 78 Domain AX is permitted to share data with Domain AY using both messaging and shared storage. The DBSy notation represents shared storage as a filing cabinet. Our other original conditions remain, that is, Domain AX is permitted to communicate with Domains AY and B but not both simultaneously. This is not captured in Figure 78 which shows maximum permitted connectivity.

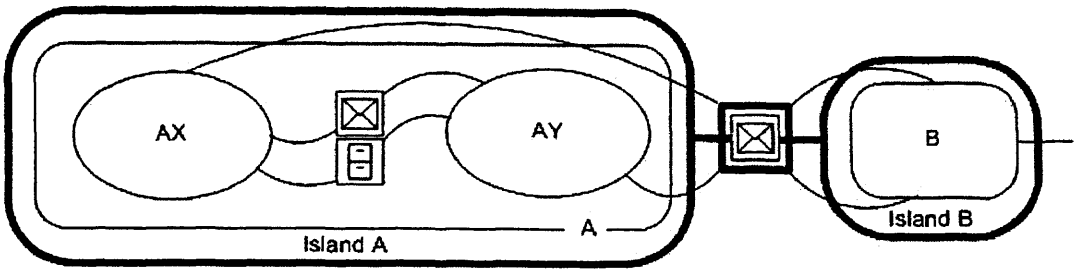


Figure 78: Use of multiple connection types

Figure 79 captures the conditional nature of the connection.

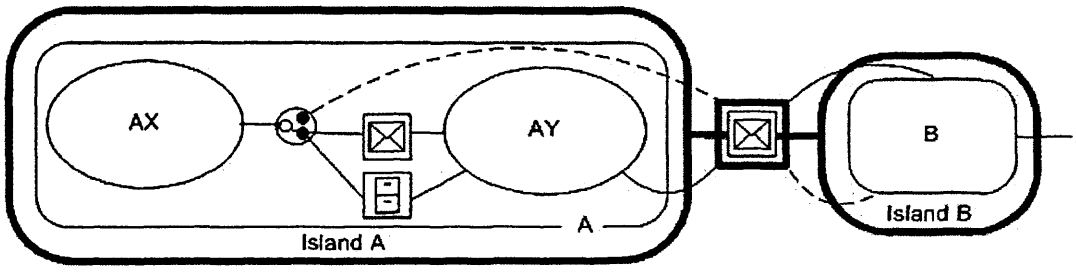


Figure 79: Multiple connection types and NPC2

In Figure 79, Domain AX has a permitted business connection with Domain AY using both messaging and filestore as a means of sharing data. The position of the conditional connector construct indicates that this path is active, that is, the baseline policy is in force. Before performing the risk assessment on this configuration, an additional case is considered. It reflects the original case study where in the event that AY is unavailable, AX is permitted to share with Domain B. However, the domain may only share via messaging. The default separation and sharing policy is maintained in the model but is of course inactive, and therefore not included in any risk assessment calculations. This model is shown in Figure 80.

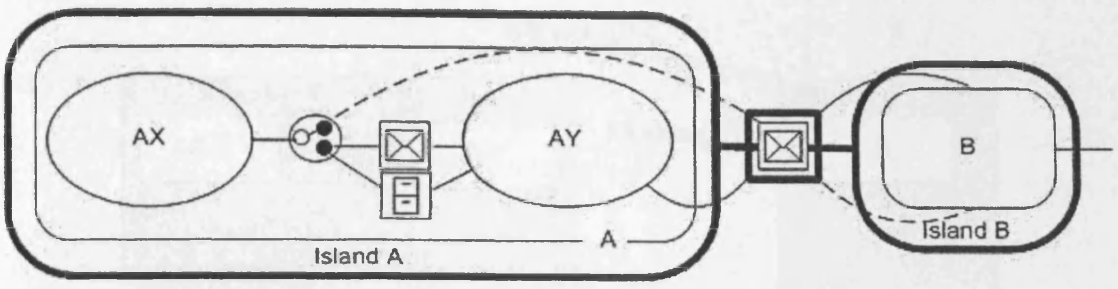


Figure 80: NPC2 permits messaging capability only

For the risk assessment process, the same procedure as described in Section 4.1 is followed. The AG and FOI are identified for the model and inbound connections from the former to the latter are isolated. During this process, only direct connections are considered. As with previous models, our concern lies with compromise paths arising from business connections, that is, we do not look at threats occurring through the infrastructure which are omitted from the figures which follow.

In the first case maximum permitted connectivity is calculated. Figure 81 shows a total of 2 compromise paths.

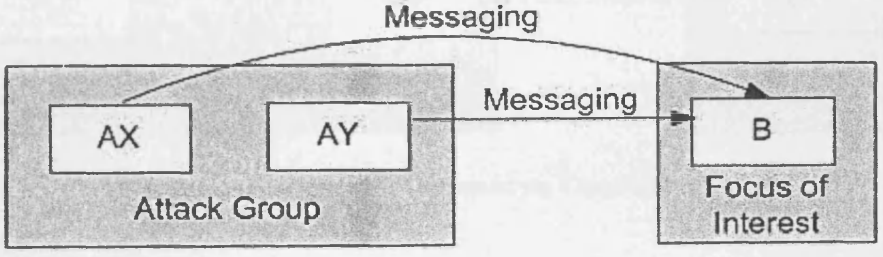


Figure 81: Compromise paths showing maximum permitted connectivity

AX and AY normally have connectivity and from the previous analysis of similar models in this thesis, it is known that an indirect compromise path between AX and B will exist as a result. For completeness the business connections between Domains AX and AY are shown in Figure 82 although their impact on the model will not become apparent until higher order connectivity is calculated. Figure 82 therefore still shows a total of 2 potential direct compromise paths.

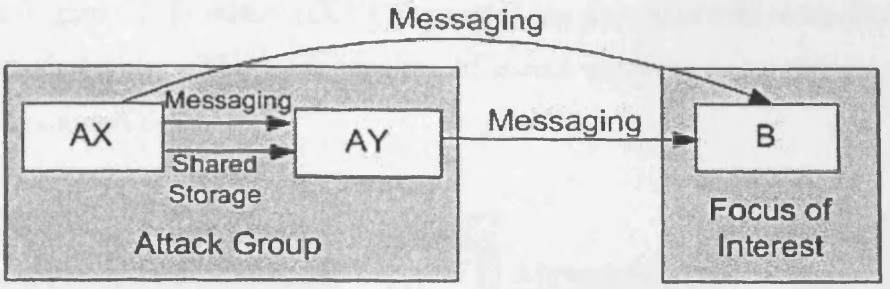


Figure 82: Compromise paths for multi-connection types; includes Domains AX and AY

In accordance with our earlier examples the models are redrawn using the conditional connector construct to add information about the temporary nature of the path between Domain AX and other domains. Figure 83 illustrates the example where AX is permitted to communicate with Domain B using messaging (as dictated by the position of the conditional connector). There are 2 potential direct compromise paths in this configuration.

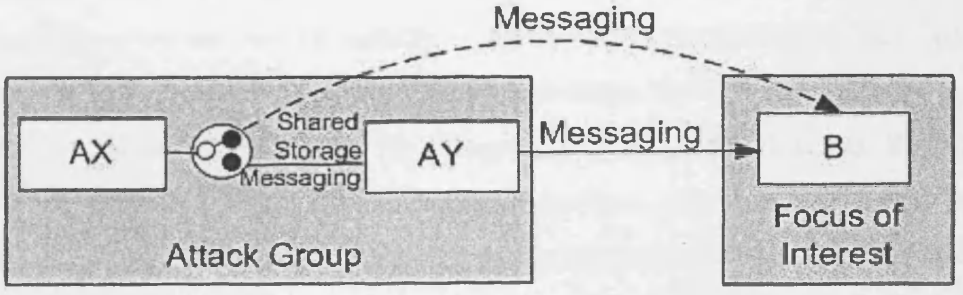


Figure 83: Compromise paths using multiple Connectors and the NPC2

The configuration in Figure 84 shows AX permitted to communicate with AY using both shared storage and messaging. There is 1 potential direct compromise path between Domains AY and B.

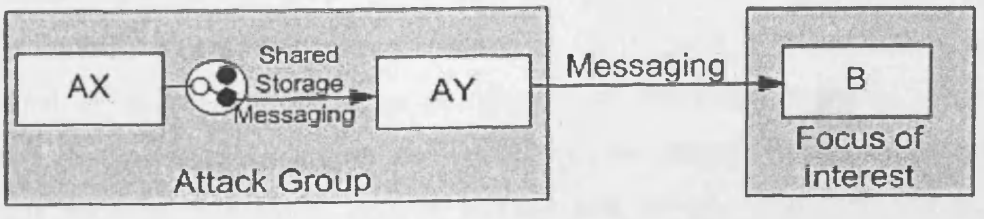


Figure 84: Compromise paths where NPC2 permits multi-connection type

Finally, in Figure 85 Domain AX is permitted to communicate with Domain AY using messaging only. The total number of direct compromise paths remains at 1 between Domains AY and B.

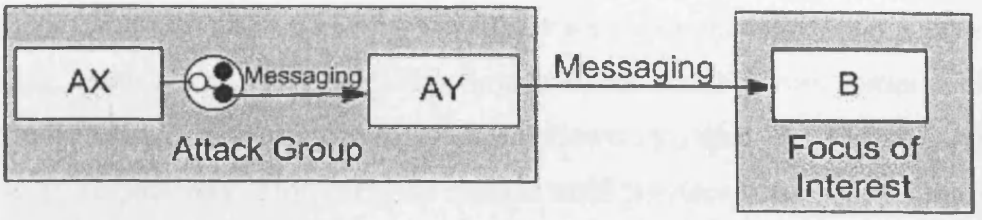


Figure 85: Compromise path where NPC2 permits messaging only

The above figures identify some difficulties with expressing multiple paths and the conditional connector. Essentially, the conditional connector is concerned with allowing a path or not and at its conception was therefore related more to the infrastructure than the connection function. Since it is now necessary to express both the existence or not or a connection *and* its function the construct requires some revision. Either, we use two constructs – one representing the selection of path, and the other the selection of function; or we combine the two capabilities into one construct (as shown currently in the diagrams). In practical terms the different selection processes will be realised in multiple devices. For example, a path could be selected via a router (where the permitted paths are predefined) or perhaps a firewall (where the permitted network addresses are inserted as rules). The ability or not to use messaging or shared storage with another domain could be defined in a number of ways; the capabilities of the user when they log on or the type of traffic that is passed through a firewall are just two examples.

One of the benefits of DBSy lies in its simplicity and uncluttered approach to documenting systems. In order to uphold this philosophy it is proposed that the InfoSec architecture models are presented as in Figures 79 and 80, where the conditional connector indicates the permitted path between domains. From this construct, the permitted business connector will be joined to appropriate sharing constructs such as messaging, shared storage and so on. The compromise path diagrams will simply use the conditional connector construct as shown in Figures 83,

84 and 85 and in accordance with DBSy guidelines, the path will be labelled to indicate the type of path and corresponding compromise that occurs between domains.

Although it is possible to address documentation details in the diagrams, when the models are represented as connectivity matrices we again encounter some interesting challenges; particularly when modelling higher order connectivity. Previously, the existence or absence of a connection has been shown in a direct or single hop matrix as '1' or '0' respectively. However, the models used previously in our calculations all considered domains as a single entity and merged all types of connections together. How do we express multiple types of connection in a matrix?

One approach is to split the instance of Domain AY into two elements: AY_1 and AY_2, which represent the messaging and shared storage capabilities respectively. The Domains are represented as $\mathbf{D}_{21}$ = [AX, AY_1, AY_2, B] for Figure 84. Connectivity is shown in C_{21}. In (65) a total of 6 potential compromise paths exist when the NPC2 permits sharing between Domains AX and AY.

$$C_{21} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \ldots (65)$$

(66) calculates indirect connectivity for the case C_{21}^2 and shows 10 compromise paths.

$$C_{21}^2 = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \ldots (66)$$

In (67) the total number of compromise paths is 16.

$$C_{21tot} = C_{21} + C_{21}^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \ldots (67)$$

By representing the two different types of connection in AY as individual elements of the same domain, we are in fact operating in harmony with the design principles of DBSy. Additional notation in DBSy includes 'cloned' domains which can be used to model a number of domains with the same security characteristics. Connections between the clone and other domains may be shown as a single element or individually depending on the information the system architect is trying to portray. A version of a connectivity matrix and the InfoSec business model may both be required in order to interpret the diagrams accurately.

C_{22} in (68) models the case where conditional policy using an NPC2 permits sharing between Domains AX and AY via messaging only. There is no permitted connection between Domains AX and B. The compromise path model for this configuration was shown in Figure 85. The Domains are represented as $\mathbf{D}_{22}$ = [AX, AY1, AY2, B]. (68) shows that the total number of direct compromise paths is reduced to 4.

$$C_{22} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \ldots (68)$$

(69) calculates C_{22}^2. The number of potential compromise paths increases from 4 to 6.

$$C_{22}^2 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \ldots (69)$$

The total for this configuration is 10 as calculated in (70)

$$C_{22tot} = C_{22} + C_{22}^2 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \ldots (70)$$

Finally C_{23} in (71) is the connectivity matrix for the case in Figure 83. Here the NPC2 permits sharing between Domains AX and B. The domains are represented by $\mathbf{D}_{23}$ = [AX, AY_1, AY_2, B]. A total of 4 potential compromise paths are given for this configuration.

$$C_{23} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \quad \ldots (71)$$

C_{23}^2 in (72) shows connectivity when the path length is equal to 2.

$$C_{23}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad \ldots (72)$$

Total connectivity is given by (73) and yields 10 potential compromise paths.

$$C_{23tot} = C_{23} + C_{23}^2 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 2 \end{bmatrix} \quad \ldots (73)$$

In summary, when another level of refinement is considered in the DBSy models and the type of connection, as opposed to the mere existence of a connection is captured, it is still possible to express the details in the InfoSec and compromise path models and connectivity matrices. It is recognised that our additional constructs, whilst adding to the completeness and accuracy of the representation, do complicate the models. The

connectivity matrices themselves could also become more complicated in cases such as C_{21} and C_{22} where individual connections are treated as independent entities within the models. To overcome this difficulty we suggest merging suitable rows and columns in the matrix. Provided there is no loss of information, this approach gives the option of viewing either a collapsed or expanded version of the connectivity matrix depending on the level of detail required by the user. To illustrate this point we take C_{21} for domains $\mathbf{D}_{21}$ = [AX, AY1, AY2, B] and merge rows and columns AY_1 and AY_2. This operation yields a 3×3 matrix structure shown in C_{24} for domains $\mathbf{D}_{24}$ = [AX, AY, B]. If the process is accurate the number and distribution of potential compromise paths in C_{24} should be the same as that shown in C_{21}.

$$C_{24} = \begin{bmatrix} 0 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \ldots (74)$$

(74) shows a total of 6 potential compromise paths with the same distribution as the results obtained in (65). Interestingly, the process of merging causes a natural departure from the use of '0' and '1' in the single-hop connectivity matrix. The content now shows the *actual* number of connections as opposed to the connectivity status of the domains. It can be stated that when $C_{ij} > 0$ a potential compromise path exists. Of course this level of detail does not necessary reveal the position of the paths, but for the system analyst, it is possible to estimate with a greater degree of certainty which domains require greater emphasis for security controls by looking at the number of actual connections.

We continue the work started in Chapter 4, which investigated the use of a Boolean representation of the NPC2 and combine this idea with the practice of merging and expanding the matrix. The same configuration as that shown in Figure 80 is used where the NPC2 permits sharing between AX and B using messaging only. b represents the *conditional* (active) path. $\overline{b}$ represents the *conditional* (inactive) path. For the purposes of calculating connectivity, $b = 1$ and $\overline{b} = 0$. The example in C_{25} yields a total of $4\overline{b} + 2b + 2$ potential compromise paths as shown in (76).

$$C_{25} = \begin{bmatrix} 0 & \bar{b} & \bar{b} & b \\ \bar{b} & 0 & 0 & 1 \\ \bar{b} & 0 & 0 & 0 \\ b & 1 & 0 & 0 \end{bmatrix} \quad \ldots (75)$$

Substitution for b and $\bar{b}$ into (76) yields a total of 4 potential compromise paths. The result is consistent with C_{23} where the connections were modelled as regular 1 or 0.

$$\begin{aligned} total &= 4\bar{b} + 2b + 2 = 2b + 2 \\ total &= 4 \end{aligned} \quad \ldots (76)$$

In C_{26} the rows and columns containing AY_1 and AY_2 are merged to reduce the size of the matrix. Like our previous example in C_{24} it is necessary to ensure that the matrix structure can be collapsed without loss of information irrespective of whether the connections are represented with a '0', '1' or a Boolean variable. (77) shows a total of $2b + 2\bar{b} + 2$ potential compromise paths and it can be seen that the integrity of the matrix has been maintained between C_{24} and C_{26}. Again, the total can be simplified to 4 by substituting for b and $\bar{b}$.

$$C_{26} = \begin{bmatrix} 0 & 2\bar{b} & b \\ 2\bar{b} & 0 & 1 \\ b & 1 & 0 \end{bmatrix} \quad \ldots (77)$$

$$\begin{aligned} total &= 4\bar{b} + 2b + 2 = 2b + 2 \\ total &= 4 \end{aligned} \quad \ldots (78)$$

To summarise, the contents of this subsection show that complexity introduced through scalability need not present an issue within the modelling process. It is possible to merge rows and columns within the matrices where the configuration lends itself to this process, that is, individual objects to be merged must be a sub-set or child of the same parent domain. This process does not lose any of the original information but includes a level of detail appropriate to the recipient. This is true not

only when the original entries in the connectivity matrices are positive integers, but also when some of them symbolise conditional connections represented by Boolean variables.

To complete this section on permitted business connections and scalability a different role for the NPC2 is considered where it is used to *remove* capability rather than enhance it. The purpose of this example is to show the capacity for the NPC2 to scale to accommodate different uses whilst preserving its overall philosophy: that of changing capability on a temporary (and conditional) basis. In our next configuration the baseline security policy allows two domains to share using both shared storage and messaging which is shown by the position of the conditional connector. The exception to this policy, as captured by the temporary business connector, is to permit sharing by means of messaging only, that is, the NPC2 indicates revocation of the ability to utilise shared storage when conditions change. This scenario is captured in Figure 86.

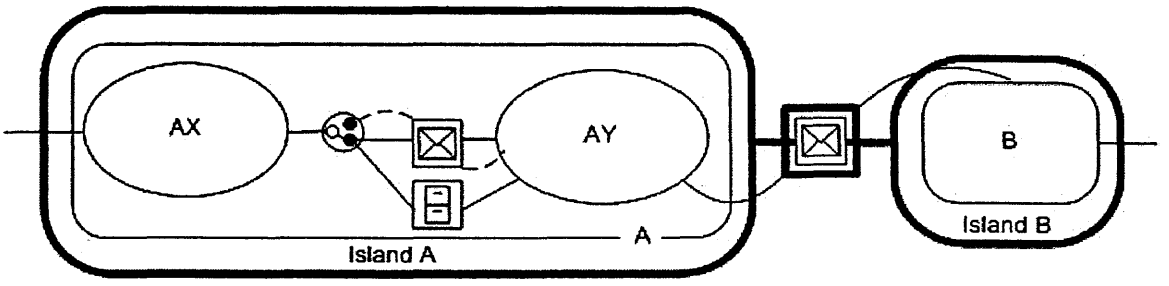


Figure 86: NPC2 used to remove permitted business connection

The InfoSec Architecture model clearly shows that in this case Domain AX actually has no direct permitted business connection with Domain B at all. Its ability to share is restricted to the agile group. As with the earlier examples, one way of differentiating between the different connections in the connectivity matrix is to treat each mode of sharing as a different entity. This is shown in C_{21} where AY_1 represents the permitted business connection between AX and AY using messaging and AY_2 is the shared storage connection. In a similar case, Figure 78, we treated the messaging connection as a sub-set of Domain AY and did not create a separate vector in the matrix to contain the status of connectivity between Domains AY and B. In C_{27} all of the connections are considered independently of one another and not as a sub-set of

the Domain. It was considered more important to experiment with the approach for this example because unlike Figure 78, in this case Domain AX *never* has any direct relationship with Domain B. We are currently undecided as to how much this affects the modelling technique and result and therefore need to consider other similar cases and make comparisons before drawing any conclusions. For now, we consider C_{27} which is able to capture and differentiate between all instances of connectivity illustrated in Figure 86 for the Domains $\mathbf{D}_{27}$ = [AX, AY, AY_1, AY_2, B] and yields 6 potential compromise paths.

$$C_{27} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \ldots (79)$$

The matrix in C_{28} shows connectivity when the NPC2 is used to revoke the business connection using shared storage (AY_2) between Domains AX and B.

$$C_{28} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \ldots (80)$$

This activity does not have any impact on the ability for Domains AY and B to share and we can see that the number of potential compromise paths is reduced to 4 reflecting the revocation of sharing using the shared storage capability between Domains AX and AY.

As a last remark on this configuration it should be noted that the model suggests that any member of Domain AX has the ability to share information with Domain B when AY is unavailable. This anomaly was noted in Chapter 3 where it is known from the original case study that not all domain members have equal status all of the time. Our

proposed solution to address the discrepancy and incorporate the dynamic dimension is of course through the NPD.

We conjecture that this process can be further enhanced if other information including probability and a risk value can be included in the models and connectivity matrices. As stated in Chapter 4, a part of the risk analysis process includes looking at the type of connection and the domains being connected in order to determine the connection level of the link and the strength of security control that is necessary to secure it. A very simplistic but nonetheless useful approach is to consider the *likelihood* of an attack using measures derived from the perceived motivation and capability of the attacker and the impact of an attack to the affected user. As described in Chapter 4, in order to illustrate the potential value of the NPC2 and NPD it is not necessary to include all possible methods by which a connection could be compromised into our assessment of risk. What is important is looking at *how* and *where* dynamic or changing security requirements can be incorporated into the process Our work is based on the notion that the ability to share varies depending on events in the environment and the NPC2 is used to show a temporary and/or conditional ability to share between two domains. The development of a fit for purpose security solution would be assisted by knowledge of the probability that a connection may be required and some indication of its impact if the path is permitted. It is expected that a value for the impact would be derived from a threat assessment process. In the defence community this could include Her Majesty's Government (HMG) InfoSec Standard 1 (IS1) [74]. Output from a threat assessment such as IS1 would take the form of a descriptive value such as 'low', 'medium' or 'high'. This could be converted to a numerical scheme representing risk in accordance with the needs of the user. This section illustrates how such values can be applied to the models and connectivity matrices to accommodate a more flexible approach to the security solution. For the purposes of our illustration it is not necessary to perform this threat assessment and in the example that follows, values for the impact weighting have been assumed. In Figure 87 the example InfoSec model includes the weighting value expressed as w_{low} and w_{high} which corresponds to the perceived level of risk. The letter b is associated with a temporary connection to show it is Boolean.

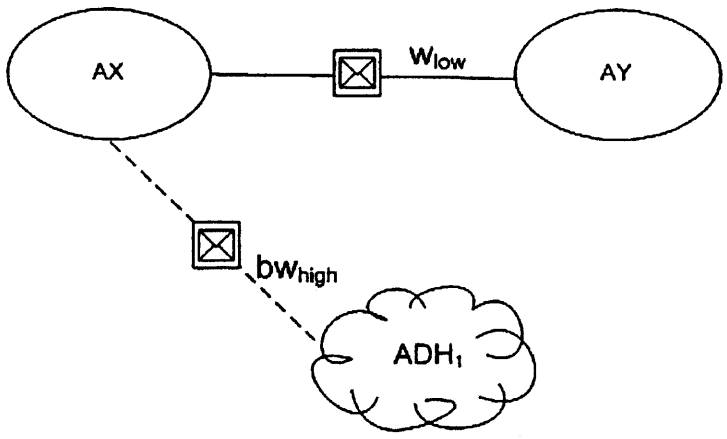


Figure 87: InfoSec Business model showing Boolean and risk weighting values

Figure 87 shows that Domain AX may, under certain conditions, share data using messaging with an ad-hoc domain as captured by the NPD. Because this capability is conditional and an exception to the baseline security policy, the temporary business connection is annotated with b to represent Boolean. Furthermore, we anticipate that the connection when active will have a high risk weighting because the ad-hoc community is untrusted and perhaps the nature of the link means that strong security controls can not be applied. By contrast the connection between Domains AX and AY is a constant and has a low risk weighting. When b =1 Domain AX is permitted to share with Domain ADH_1 in addition to AY. Connectivity for this configuration is given in C_{29} for the Domain configuration $\mathbf{D}_{29}$ = [AX, ADH_1, AY].

$$C_{29} = \begin{bmatrix} 0 & w_{low} & bw_{high} \\ w_{low} & 0 & 0 \\ bw_{high} & 0 & 0 \end{bmatrix} \quad \ldots (81)$$

By using this augmented notation in the matrices it is possible to add more information about the nature of the potential compromise path which could be used in the security planning process. For example in C_{29} the presence of the conditional connection between AX and ADH_1 completely changes the risk profile of the system. In this state, ADH_1 presents a risk to AX but more importantly, it also presents an indirect risk to Domain AY with whom it is not permitted to share data. This is visible in (82) for the case C_{29}^2.

$$C_{29}^{2} = \begin{bmatrix} w_{low}^{2} bw_{high}^{2} & 0 & 0 \\ 0 & w_{low}^{2} & bw_{low} w_{high} \\ 0 & bw_{low} w_{high} & bw_{high}^{2} \end{bmatrix} \quad \ldots (82)$$

(83) calculates the total risk weighting for this case.

$$C_{29tot} = C_{29} + C_{29}^{2} = \begin{bmatrix} w_{low}^{2} bw_{high}^{2} & w_{low} & bw_{high} \\ w_{low} & w_{low}^{2} & bw_{low} w_{high} \\ bw_{high} & bw_{low} w_{high} & bw_{high}^{2} \end{bmatrix} \quad \ldots (83)$$

The summation of (83) is $2(w_{low} + w_{low}^{2}) + 2b(w_{high} + w_{high}^{2} + w_{low} w_{high})$

The results show that the level of complexity in the models increases rapidly when higher order connectivity is calculated. It would not be practical to calculate such cases manually, but using software developed in MATLAB [43] the content can be simplified to improve the legibility and usefulness to the analyst. A less cursory glance at (83) gives a clearer indication that the potential risk to Domain AY would be very high when the Boolean is true and connectivity is permitted with the ad-hoc Domain ADH_1. It is also very clear which part of the risk is the inevitable result of permitting a business connection and which element can be controlled through the Boolean permitting the NPD connection.

To complete this section some preliminary studies are performed on the combined use of a probability value with the Boolean, that is, a measure of how *likely* the activation of this connection will be. Within an agile environment it is assumed that this value will itself change over time and in accordance in events but nonetheless, an initial estimation of the probability yields some useful results and enlightens us regarding the potential usefulness of this approach. The probability is given as p_{low}, p_{high} where $0 < p_{low}, p_{high} < 1$.

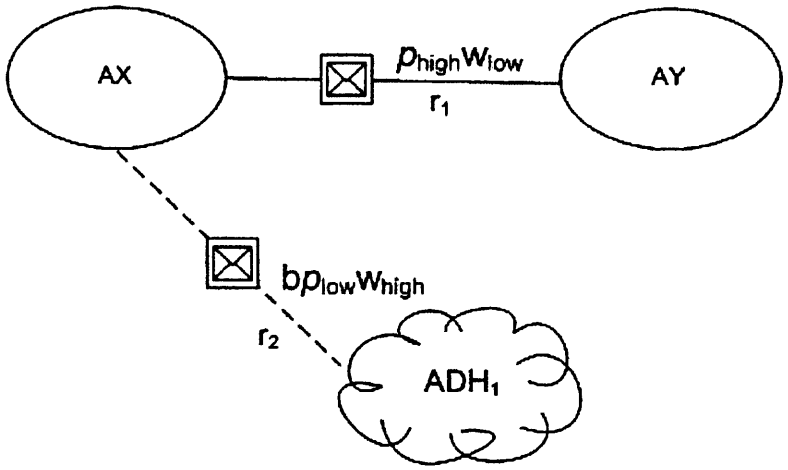


Figure 88: InfoSec business model showing probability, risk weighting and Boolean values

Figure 88 shows that the probability of the conditional connection between Domains AX and ADH_1 is low. However, when the connection is active the risk weighting is high. Conversely, the probability of connectivity between Domains AX and AY is high but the risk is low.

In Figure 88 the risk value for each connection may be expressed as r_1 and r_2 where:

$$r_1 = p_{high} w_{low} \quad \ldots (84)$$

$$r_2 = bp_{low} w_{high} \quad \ldots (85)$$

(86) presents the matrix C_{30} for Figure 88, where domains $\mathbf{D}_{30}$ = [AX, AY, ADH_1].

$$C_{30} = \begin{bmatrix} 0 & p_{high} w_{low} & \bar{b} p_{low} w_{high} \\ p_{high} w_{low} & 0 & 0 \\ \bar{b} p_{low} w_{high} & 0 & 0 \end{bmatrix} \quad \ldots (86)$$

The vulnerability of this configuration is defined as the total of the elements in the matrix. (87) calculates the vulnerability for C_{30}.

$$V_{30} = 2p_{high} w_{low} + 2\bar{b} p_{low} w_{high} = 2(r_1 + \bar{b} r_2) \quad \ldots (87)$$

As shown in (87) the equation can be simplified using the *r*-notation in (84) and (85).

It is known from Chapter 4 that $\bar{b}$ represents an inactive conditional connector therefore the results in (87) confirm that when connectivity between Domains AX and ADH_1 is not permitted, connectivity and thus potential compromise paths, are quite low. However, when the conditional connector is active as shown by b in the matrices, the results shown in (88) are obtained. For the purposes of calculating connectivity, $b = 1$.

$$C_{31} = \begin{bmatrix} 0 & p_{high} w_{low} & bp_{low} w_{high} \\ p_{high} w_{low} & 0 & 0 \\ bp_{low} w_{high} & 0 & 0 \end{bmatrix} \quad \ldots (88)$$

Vulnerability for (88) is calculated as the sum of all elements in C_{31}. The results are given in (89) where we also simplify using the *r*-notation of (84) and (85).

$$V_{31} = 2p_{high} w_{low} + 2bp_{low} w_{high} = 2(r_1 + br_2) \quad \ldots (89)$$

The permitted connection between AX and ADH_1 presents an indirect compromise path to Domain AY which is clearly when C_{31}^2 is calculated in (90).

$$C_{31}^2 = \begin{bmatrix} p_{high} w_{low}^{\ 2} + bp_{low} w_{high}^{\ 2} & 0 & 0 \\ 0 & p_{high} w_{low}^{\ 2} & bp_{high} w_{low} p_{low} w_{high} \\ 0 & bp_{high} w_{low} p_{low} w_{high} & bp_{low} w_{high}^{\ 2} \end{bmatrix} \quad \ldots (90)$$

Total connectivity for $C_{31} + C_{31}^2$ is given in C_{31tot} and shown in (91).

$$C_{31tot} = \begin{bmatrix} p_{high} w_{low}^{\ 2} + bp_{low} w_{high}^{\ 2} & p_{high} w_{low} & bp_{low} w_{high} \\ p_{high} w_{low} & p_{high} w_{low}^{\ 2} & bp_{high} w_{low} p_{low} w_{high} \\ bp_{low} w_{high} & bp_{high} w_{low} p_{low} w_{high} & bp_{low} w_{high}^{\ 2} \end{bmatrix} \quad \ldots (91)$$

The vulnerability of the configuration of (91) is cumbersome but can be simplified in (92) by using the *r*-notation in equations (84) and (85)

$$V_{31tot} = 2[r_1 + r_1^2 + b(r_2 + r_1 r_2 + r_2^2)] \quad \ldots (92)$$

(92) shows that the proliferation of connections and correspondingly increased level of risk from the additional compromise paths when the conditional connection, the NPD, is permitted. This is the element of (92) prefixed by *b*.

$p_{low}, p_{high}, w_{low}, w_{high}$ may assume any value in a range defined specifically for a scenario. For example, as a result of an earlier threat assessment process, it may be considered that the weighting of the connection defined as w_{low}, w_{high} may range from 1 – 10 where 1 represents the lowest weighting and 10 the highest. Furthermore, observation of network behaviour and connectivity may lead the system designer to conclude that the probability of a connection being active is given as p_{low}, p_{high} where $0 < p_{low}, p_{high} < 1$. These values can be used to further illustrate the effect of applying the NPD concept. We take the case where $p_1 = 0.1$ and $p_{high} = 0.9$ and $w_{low} = 2$ and $w_{high} = 7$. Substitution of these values into (84) and (85) enables us to calculate the cost of each link in Figure 88. In this case, r_1 which connects Domains AX and AY has a cost of 1.8 and r_2, the conditional connection between AX and the NPD ADH_1, has a cost of 0.7. Numerically, the conditional link is the lower risk link in this case. The values for r_1 and r_2 can be used in (93) and (94) to calculate the *actual* cost to each domain (in terms of risk) of permitting each connection. The results of substituting $r_1 = 1.8$ and $r_2 = 0.7$ into (87) is shown in (93)

$$2[(1.8) + \bar{b}(0.7)] = 3.6 + 1.4\bar{b} \quad \ldots (93)$$

Since $\bar{b} = 0$ the vulnerability in this case is $2r_1$ which is 3.6.

The results of substitution into (89) are shown in (94).

$$2[(1.8)+b(0.7)]=5. \quad \ldots (94)$$

In (94) the NPD is permitted therefore $b=1$. This increases the level of vulnerability from 3.6 in (93) to 5 in (94) a difference of 1.4. The Boolean values are left quite deliberately in (93) and (94) because of the significant visual impact. The security architect can immediately see the affect of permitting the temporary connection and the contribution that each link adds to the total. The calculations in (93) and (94) are for direct connections only. In (92) the vulnerability arising from indirect (2-hop) connectivity was calculated because it was recognised that the NPD, when active, would impact Domain AY. The result of substituting the probability and weighting values into (92) is shown in (95).

$$2[5.04+b(2.45)]=10.08+4.9b. \quad \ldots (95)$$

When $b=1$, total vulnerability for (95) will be 14.98. When $b=0$, that is the NPD is inactive, total vulnerability will be 10.08.

Again, we have separated the Boolean element of the equation because it allows us to differentiate between the *minimum* risk, which excludes the Boolean element and the *additional* cost of permitting the NPD. In the case of (95) minimum risk is 10.08. The NPD would contribute an additional 4.9 when permitted - an increase of about 50%.

The example described above demonstrates very clearly the effect of permitting the NPD. The minimum risk is inevitable and arises from the permitted business connection between Domains AX and AY. However, the system architect has some control over the maximum risk since it is known from (94) and (95) how much additional risk will be incurred if and when the Boolean element is activated. Armed with this information stakeholders can make more informed decisions about the conditions under which the connection may be permitted and its use. The security policy may state for example, that sensitive data may only be sent to AY because the weighting of the link is low. However, because its availability is high this could potentially give an attacker greater opportunity to intercept or otherwise breach the security of the link. The 'high' weighted but conditional link may, in some instances,

be at least as secure because the probability of it being active is low and there is only a small window of opportunity for the attacker to become aware and attempt to subvert the connection. In Figure 88 only one part of the NPC2; the conditional business connector, was applied. Given the results in (93), (94) and (95) it may be desirable to utilise the conditional connector element of the NPC2 to only permit one connection or the other between Domains AX , ADH_1 and AY. Under this regime the potential for a high risk and low risk link to be active simultaneously is removed.

7.1.4 Preliminary Conclusions

The examples given in this section are not overly complex but do provide sufficient evidence to allow us to form some preliminary conclusions. The NPC2 has the ability to scale to larger configurations involving different sets of possibly related conditional connections between domains. Furthermore, it can also scale to include additional functionality. Although it was first conceived to show a temporary change in the separation and sharing profile and be used to elevate a subject's capabilities, the NPC2 could also be used to indicate a temporary change which results in the *revocation* of the ability to share.

Although our additional constructs complicate the DBSy models they do add benefit by enhancing the comprehensiveness and veracity of the models. Scalability also results in greater complexity within the connectivity matrices. However, we have established a method to address this by merging vectors within the single-hop connectivity matrix where appropriate and possible. This work resulted in the use of values other than '0' and '1' since we were looking at *numbers* of connections as opposed to the existence (or not) or a connection. By moving away from our earlier approach we conceived the idea of representing the NPC2 state as a Boolean variable. This approach shows that the *potential* for connectivity and removes the need to consult a separate and additional matrix.

Finally our work combines the inclusion of probability and risk weighting values alongside positive integers and Boolean variables in a single matrix. In [40] Hayat *et al* proposed the use of a risk ranking system based on the severity of the risk which could be used to assist with the placement of security controls. Unlike Hayat *et al,* our

assessment of risk considers the *conditional* properties of the link in addition to a risk weighting value. In the matrices we include the probability of a conditional connection being *active* (and therefore permitted) or not. This can have a profound effect on the risk profile of the system and the subsequent design of the security solution. Whilst it is unusual to mix real and Boolean variables in a matrix, provided care is taken, the work in this chapter shows that the approach we have adopted can significantly increase the utility of the matrices and comprehensiveness of information presented to the system analyst.

Although some potential difficulties have been identified when the NPC2 and NPD are scaled, we maintain the view that by capturing change within the InfoSec models and connectivity matrices the system designer is given 'advance warning' about security issues which can be used to influence the design of the system. Some problems may be overcome through technology as part of the implementation or in other cases this improved level of information could be used to influence procedures governing the way in which the system is operated.

7.2 Scaling the Non-Persistent Capability Concept in Implementation

The previous sub-sections described the use of more complex scenarios and the impact on use of the NPC2 as part of the modelling and risk analysis process. The models showed that it is possible to capture increasingly complicated conditional requirements. However, the ability for the NPC2 implementation architecture to accommodate growth without becoming unusable is a matter for further investigation. It should be noted that within the time constraints for this research it has not been possible to physically construct larger configurations involving multiple NPC2 installations: therefore this work would form part of our future research. We are fully aware of the challenges that could arise from scaling the implementation. The problems are described in this section, accompanied by proposed solutions.

In order to assess whether the NPC2 can physically scale certain questions need to be addressed including, for example, how closely coupled the NPC2s are with one another. If they are tightly coupled for instance, then the impact of each NPC2 must be considered within the context of other devices in the system. This could quickly

become unmanageable for the system designer in assessing all possible interactions and interrelationships between elements of the system. Secondly, it is necessary to consider the extent to which we could or indeed should reuse components within the NPC2 architecture to perform other tasks. If each NPC2 is unique in function, adding other NPC2s into the system could result in a management and development overhead that could render development impractical. Finally, our notion of fit for purpose security is aimed at eliciting the smallest change possible to achieve the desired change to the security profile of the system. The NPC2 in our demonstration example is used to modify the capability of one subject, the 2IC, when conditions necessitated this change. In the small agile groups our work targets, it is quite probable that the number of subjects affected by a change will be small. However, we recognise that in order for our solution to be scalable it is necessary to consider the situation where changes to security will affect a larger number of subjects in the system. There are of course a number of ways in which this can be achieved without affecting the core components of the NPC2. This aspect of scalability, along with the coupling and reuse of components are all addressed in the sub-sections that follow.

7.2.1 The Non-Persistent Capability Concept and Multiple Subjects

It is entirely feasible that a single event triggered by the Environment Monitor could cause multiple refinements to the security policy. It is also the case that the security policy itself can be associated with many users who will all be affected by the changes. This association could be on a 'per user' basis although a more popular and manageable approach is to employ the use of Role Based Access Control (RBAC) [75]. In this model a role representing a job function in the organisation can be occupied by one or multiple users. Access permissions assigned to the role are automatically inherited by all occupants. The amount of overhead for system administrators is thus reduced since a single change made to the permissions of a role potentially affects a large number of users. Of course no solution is perfect, and the role itself must be carefully managed so that permissions are actually appropriate for all of the occupants. Where the number of users becomes large, it is possible to get disparity between some of the occupants and their capabilities. As shown in Figure 57, Chapter 5, the security policy component belongs in part to the NPC2 and in part to the external environment. Effectively it is managed by the standard system but

manipulated by the NPC2. We conjecture that provided the security policy sub-system is sufficiently flexible to allow us to assign a (modified) security policy to one or many users without the system becoming unmanageable then we do not foresee any major issues that would prevent us from scaling the NPC2 to multiple subjects.

SELinux was used for our demonstration environment. As described in the literature review, SELinux is based on type enforcement technology where a domain type is assigned to all objects on the system. This is not necessarily a one-to-one relationship as used in our design. Multiple objects can belong to the same type. Under this regime, type enforcement behaves in a similar manner to RBAC since it is possible to affect many associated objects with one permission change. In future work it would be possible for us to extend the reach of our security policy and create an example where it is appropriate to label a number of objects with the same domain type. We could then confirm, by checking through application log files, that when the security policy change is triggered through normal operation of the NPC2, all of the associated objects receive the permission change.

7.2.2 Coupling of Non-Persistent Capability Concept Components

The extent to which NPC2 components are dependent on each other largely depends on the requirements of the system. For example in Figure 73 and the connectivity matrix C_{15} some interesting properties were observed when the security policy requirements were modelled. Recall that Domain AY was not permitted a business connection with both Domains B and C simultaneously and the NPC2 correctly showed this exclusive condition with the position of the switch. However, Domain B was also controlled by an NPC2 that did not permit simultaneous business connections between Domains C and AY and itself. We had the case where Domain AY could quite legitimately hold a permitted business connection with Domain B; however, unknown to Domain AY, Domain B also had a perfectly legitimate business connection with Domain C. On the face of it this contravenes the security policy because effectively Domain B would appear to have a connection with both. In Section 7.11 we proposed that the modelling process used was possibly too coarse to capture the level of detail in such a situation. However, it fulfilled a need in that the

system designer is alerted to a potential issue that should be resolved through the implementation. So, how might we approach a solution?

Essentially, this is a problem of conflict resolution where two different and legitimate sets of conditions occurring at the same time, result in a potentially illegal operation. We contend that the NPC2s do not necessarily need to be aware of each other's rule sets, only their own behaviour. The NPC2 controlling whether Domain B is permitted a business connection with Domains AY and C would verify outstanding capabilities before permitting sharing to take place. This situation is a standard task for the Reference Monitor component within the security sub-system where the security policy protecting an object is checked when an access request is made by the subject. The policy decision is taken and enforced dependent on the outcome of this verification process.

The ability to resolve policy conflict is very much dependent on the expressiveness of the underlying security policy language. As the complexity of the system increases so it becomes correspondingly more difficult to account for *all* of the interactions within the system. In our example, when the read process requested access to the message queue the security policy protecting this object was verified. All of the time the Boolean variable was set to false, the request was denied and this access decision enforced. This is an area which would definitely require further exploration as part of future work because the ability to resolve multiple conflicts using the NPC2 is unknown.

There are also potential issues which originate from the communication infrastructure and environment itself. If the communications protocol used for messaging requires a response from the destination device, this low level activity may be in conflict with the security policy. We provide an example in Section 7.1.1 Figure 74 where the InfoSec business model shows a one-way communication between the Low and High-1 Domains. In this configuration messages originating from the Low Domain cannot be acknowledged by the High-1 Domain because this will contravene the security policy. This is not necessarily a scalability issue, but just as we encountered with our demonstration example, the case does provide another example where the system designer has to be fully aware of the environmental factors that will both influence

and be influenced by the InfoSec models. Of course in such a situation it would be necessary to find other solutions. If the passing of messages between Domains Low and High-1 was critical for example, then we may elect to send the message a number of times to increase the chances of it being received since we are unable to confirm reliability by any other means. This is not an efficient use of network bandwidth but is an option nonetheless.

7.2.3 Reusability

The NPC2 has deliberately been designed so that each component can act independently if necessary and be configured to detect and manage other events. The set-up itself is therefore implementation specific although the core codebase should be applicable for any implementation environment with minimal change. Although the elements of the NPC2 are being used to deliver a flexible security solution, only some of these elements are part of the actual security mechanism itself. The Environment Monitor and Event Manager are designed to interact with the resident security system to initiate a change in the policy rules on detection of the pre-defined condition(s). They perform an essential role in the decision making process and are responsible for triggering the change in security policy. This is why the communication between each component and the resident security system itself must be trustworthy. The resident security system could consist of any technology provided it presents the appropriate interfaces to our NPC2 applications to enable manipulation of the underlying rules. In our solution we have elected to use SELinux because this provides the open development environment and flexibility that we require. Had a different technology been adopted, development of the NPC2 interfaces would have been quite different.

The Environment Monitor in our demonstration system is a one-to-one relationship, that is, the software is monitoring the environment for a specific behaviour that will be compared against our threshold and trigger the policy change. However, as shown in our architecture in Chapter 5 there is an opportunity to extend this concept. We propose this could work as follows:

- Many events are monitored and managed by an overall event manager. Each event triggers a different policy change and thus a different capability.

- Many events could be fused by the overall event manager. The result of this fusion process could be a single or multiple policy change(s).

- Some combination of the above where multiple events (but not all) are fused and others are managed independently to result in multiple policy changes.

Quite clearly this model introduces significant complexity in terms of development. The Event Manager needs a certain degree of 'intelligence' to process different events. However, it has the potential to be a highly configurable middleware solution with the opportunity to 'plug in' any type of Environment Monitor within the Event Manager.

7.3 Summary of Scalability

This chapter can be summarised as follows. The main emphasis of our work has been directed towards scalability and the NPC2 and NPD concept within the modelling and risk analysis process. The notation can be used in larger and more complex configurations using multiple NPC2 constructs or in the case of the NPD, multiple connections with existing domains. Moreover, the NPC2 can be used to capture different *types* of connection and show where their use is conditional and temporary. In all of these cases we believe that the models have not become over complicated nor have they lost the clarity of expression which is a desirable characteristic of this graphical approach. System designers are presented with a more informed view of the system under different states and through the use of the connectivity matrices can easily calculate the potential number of compromise paths when changes to the separation and sharing profile become necessary. The risk analysis process identified potential issues with the matrix calculations in that they can become large and possibly unwieldy. Our solution is to merge elements of the matrix where appropriate. Sub-domains of a parent or what we term 'child' domains can be combined to provide a condensed view of the matrix without loss of information. This approach can be utilised irrespective of how connectivity is represented in the matrices. Finally, whilst we have not had the opportunity to progress larger models to the implementation

phase, some of the challenges have been outlined that we believe should be addressed if multiple NPC2s are distributed in a system or the functionality of a single NPC2 is extended.

From the work described in this section we conclude that sufficient evidence has been collected to support the claim that the concept of the NPC2 and NPD will scale, at least in the modelling process. Theoretically we believe scalability of the NPC2 and implementation is sound. However this has yet to be confirmed through practical experimentation.

Chapter 8: Conclusions and Further Work

The purpose of this final chapter is to reflect on the programme of research and in doing so, draw conclusions regarding the success of the study and present opportunities for further work. This raises an interesting question. Our research deals with concepts inside of a complex and broad subject space. As the work has evolved it has naturally become more qualitative than quantitative although established mathematical techniques have been applied through our use of matrices to model connectivity matrices in Chapters 4 and 7. The discursive nature of the approach is a natural consequence of the psychology and working preferences of the author, but we believe is also appropriate to the research area. So, in a study such as this how do we measure and communicate the success of our work? In order to answer this question we return to the original research problem and our work, documented in this thesis, which has been carried out to solve it.

In 2.1 the research problem asks "...what options are available for engineering solutions appropriate to the Network Enabled Capability military initiative?..." Our background research led us to conclude that a more radical approach to security was required. Furthermore, we proposed that the answer lay in developing *fit for purpose* security which recognises that in contemporary military operational environments, security must adapt and respond to the changing needs of an organisation, whilst still meeting requirements for appropriately high levels of assurance. Furthermore, the military tend to operate in hostile and resource-constrained environments where changes to the security of the system may quickly become a management overhead and unnecessary distraction for the user. This provided another motivation for our research where fit for purpose security could also be viewed in terms of robustness. In accordance with Ashby's law of requisite variety [5], a system with many options is better equipped to deal with change. Therefore if we engineer the opportunity for the system to adopt a different security policy under particular circumstances, we conjecture that not only may it survive for longer without the need for human intervention it will also provide security that is fit for purpose when required.

As described in Chapter 1 we proposed that fit for purpose security involved manipulating a secure platform in a way that could be understood whilst affording the opportunity to modify the security profile in accordance with user requirements. Measuring the impact of change involved having the means to describe and model updates to the security policy so that the effect of flexibility could be understood and risk mitigated where necessary. We proposed that assessing security under different states afforded the designer an opportunity to select *appropriate* security for that state. This can increase both efficiency and effectiveness of the controls. Security requires the synthesis of security policy, architecture and mechanism. Therefore, it was necessary to examine each of these areas and determine how flexibility could be described and verified within the security policy and then practically engineered through architecture and mechanism. This holistic approach enabled us to develop a methodology for providing the necessary flexibility.

Early on it became obvious that the key was security policy; however, this would only provide one component in a much wider solution. Flexibility requires a degree of intelligence which involves detecting and acting upon an event of interest in the environment. In our case, the action is a change to a user's capability controlled by the security policy. Each element within this chain depends on the others to deliver the overall fit for purpose vision. Our solution to this problem lies in an intelligent connector, the NPC2. In Chapter 3 we showed how the NPC2 could be represented in high-level graphical models such as DBSy through an addition to existing notation. In this role the NPC2 captures temporary and conditional permitted business connections that *may* be required when certain conditions prevail in the external environment.

A fundamental part of the fit for purpose security vision is the provision of the ability to reason about the level of potential risk that flexibility introduces into the system. In Chapter 4 it is shown that the NPC2 provides this ability: examples are given demonstrating how the NPC2 can be incorporated into the risk analysis process to calculate the effects of modifying the separation and sharing profile of a system. In an attempt to employ analytical techniques that are formalisable – and therefore potentially certifiable – connectivity matrices are used to model the way that sharing can take place between different domains in the system. In both Chapters 4 and 7 it was shown that the matrix elements are not restricted to a '1' or '0' (representing the

presence or absence of a connection). In Chapter 4 Boolean values (representing the conditional nature of the NPC2 connection, were used. In Chapter 7 the ideas were extended to include numerical probability values (representing the reliability of particular communication links), or numerical penalty values (reflecting a weighting of the risks associated with particular sharing arrangements) in the matrices. In total all 4 representations were used simultaneously and provided care is taken with the mathematics, this approach results in an enhanced view of the potential and actual risk to a system under dynamically changing conditions. Moreover, the results have a strong visual impact. The system analyst is able, in certain cases, to easily identify the minimum and maximum risk and the contribution made by the non-persistent constructs. We conclude that this level of detail has unprecedented benefits to system stakeholders in the design and execution of fit for purpose security.

In addition to demonstrating ways in which the NPC2 can provide the conceptual basis for flexible security, our work has examined ways in which the concept can be implemented and engineered into a working solution. In Chapter 5 a modular and technology independent architecture was presented which describes the components required to construct the NPC2. As stated previously, although absolutely fundamental and central to flexible security, the security policy itself is one part of a wider strategy in the engineering solution. Ultimately the policy is used to manipulate the underlying security controls and infrastructure because it contains the rules which are implemented by the security enforcement agents. However, the change process needs to be triggered. Given that one of our design objectives was robustness we also sought to build an autonomous system which removed the human from the loop. Chapter 5 shows the security policy encompassed by processes to monitor the environment (for the event of interest), manage the observed event, and trigger the change in security policy. From this point, the security policy itself is responsible for instantiating a new rule set which gives the change in permitted (user) capability. The NPC2 architecture describes the functionality required from individual components in the stack. Chapter 6 documents an example implementation hosted on the SELinux platform which proves that the objective of the NPC2 and the proposed architecture can be physically engineered and shown to work. Although small-scale, the example successfully demonstrated each of the NPC2 elements working together to achieve an autonomous change in security. As a result, the subject controlled by the security

policy was permitted to share data it was previously separated from. Chapter 7 presented some of the challenges that would need to be addressed in order to scale the implementation in larger and more complex scenarios. Since there was not an opportunity to pursue this area of work, it clearly presents an opportunity for further study and possibly an opening for collaboration with other parties.

The research has been taken a stage further by the introduction of another new concept, the NPD aimed at modelling ad-hoc collaboration activities and their potential impact on risk. Although it has not yet been carried through to implementation, the NPD has been shown to be useful in high-level modelling and risk analysis using the same test environment as the NPC2.

During the course of the research a number of problems have been identified which remain unsolved. The research question embraces a vast and complex subject area and there was neither time or in some cases the expertise to address all of the issues as part of this programme. To complete this chapter we present some of the challenges which we believe are worthy of further investigation and will contribute to this interesting and useful area of work.

At the presentation of [24] the question of using the NPC2 in domains other than the military was raised. It is recognised that the defence environment is very demanding because of the requirement to provide 'provable' security. Quite rightly any notion of flexibility is likely to be regarded with reticence. In a commercial environment the NPC2 would be equally useful and could probably be integrated more readily because there are less stringent needs for high assurance. During the research we have been mindful of but not driven by other markets because it appeared that if the concept could be successfully demonstrated in a military context, other business sectors should, in theory, offer significant opportunities and potentially a less demanding development environment. At the time of writing an up-lift to [24] is in progress for submission to the Journal of Information Warfare [76]. In this version we consider the NPC2 in the context of other work [77] which targets adaptive access control in a hypothetical financial brokering system. From the description of the requirements it is apparent that the NPC2 could be applied in this application domain. The work of Cheng *et al* in [77] is principally concerned with the classification of resources and

having adjustable 'hard' and 'soft' boundaries between classifications based on the calculated level of risk. By comparison, the NPC2 is designed to change user capability rather than the classification of a resource. Nonetheless, in a scenario such as the brokerage system it could operate in either capacity. The NPC2 would normally work with the underlying security sub-system to modify security permissions. However, provided the functionality was in place, it could equally be used to reclassify objects and deny/permit access through this mechanism. A preliminary and cursory look at other application domains indicates that the NPC2 has wider application. One of our recommendations is that the concept is exploited not only in the military environment but also in potentially more accessible civilian and commercial domains.

As a result of our collaborative research with De Montfort University we concluded that it would be useful to combine a graphical and intuitive modelling approach such as DBSy with a formal security policy framework such as SANTA and that furthermore, this conjunction would widen the appeal of both approaches to different stakeholder communities. One of the disadvantages of SANTA is that it requires significant expertise of the security policy language to express high level security requirements. However, as described in [22] the power and expressiveness of the language was absolutely fundamental for capturing and refining changing security requirements such as those in our case study. The DBSy models provided an intuitive means of capturing high-level separation and sharing requirements; although, if these models are to be useful in the development of implementable policy, it is necessary to refine the connections into a series of potential interactions, stakeholders and observable attributes. It is recognised that DBSy was not intended as a security policy language. Nonetheless, we believe its value can be extended if we can attach a level of meaning by underpinning the models with a formal language and possibly automate the creation of a formal policy. This also presents additional opportunities for verifying the security policy. System stakeholders will be able to perform a visual check of the system and its potential compromise paths. This may be complemented by a more structured analysis using the connectivity matrices and a formal analysis of the model using tools from DeMontfort University such as SPAT. This potential has been shown in our work where both approaches have been applied independently. Our

conclusion is that additional work to combine the two could yield significant benefit to system developers and system owners alike.

In Chapter 7 we began to explore the opportunities for using risk weightings and probability values in our calculation of potential compromise paths. This concept is not new; we know for example that Hayat *et al* proposed the use of a risk ranking system in [40] which is based on the severity of the risk. However, what distinguishes our approach from others is our conjunction of weighting values with the notion of *temporary* and *conditional* connectivity provided by the NPC2 and NPD constructs. This approach allows us to calculate the potential effect of permitting a change in capability and offers an unprecedented level of useful information to feed into the decision making process which could include a risk ranking system such as that from Hayat *et al* [40]. Our practical implementation of the NPC2 showed it was possible to create a system which autonomously changes capability and the security profile. We suggest that output from the risk and probability calculations could be used by an Environment Monitor as part of its decision making process. Armed with this information and other parameters, the decision could be taken to permit (or deny) sharing to take place without necessarily involving the human user. Although there is still a lot of work to be done in the area of risk assessment and flexibility, we propose that extensions to our ideas which utilise risk and probability metrics in addition to the Boolean representation of a connection would provide valuable insight into designing secure systems in the future. Finally, the NPC2 and NPD can both have a significant influence on the proliferation of connections in a network and our view of connectivity growth patterns. We believe this may challenge existing theories in this area such as those espoused by Metcalfe [46] and Briscoe *et al* [47]. We currently have a paper in production which explores this interest further and will be published separately from this thesis.

Although our implementation of the NPC2 was successful, it is recognised that numerous opportunities exist for augmenting the prototype system either using the same technology as that in our example or completely different technology. Once we had shown fit for purpose security working through practical implementation of the NPC2, we were not at liberty to explore other aspects of the case study that were deemed to be of interest. For example, given that the devices are operating in an agile

mission group it is possible for them to all contribute to the environment monitoring process. Our prototype was deliberately small-scale and relied on one member, the 2IC, to gather intelligence. It would be useful to extend the study and investigate how the collaboration process between all members could be engineered and processed using the NPC2. In theory our research leads us to conclude that it is possible; indeed Chapters 5 and 7 describe ways in which the NPC2 could be scaled in larger configurations. One of the proposals included using multiple inputs to the environmental monitoring and event management processes. Furthermore, we could trigger many changes to the security policy affecting one or multiple users. All of these activities would result in additional capability and interest to the practical usage of the NPC2.

The prototype solution of the NPC2 was hosted on and used the security capabilities of SELinux. This is feature rich and offers many possibilities for extending the functionality of the NPC2. For example, we elected to show a temporary change in a subject's authorisation on the system. This approach was quite deliberate since the objective was to cause minimal disruption and change to the security profile of the system. SELinux supports features such as labelling network packets. Some work has already been done on combining this capability with security policy to restrict the processing of packets to those of a pre-defined type [78]. We see an opportunity to combine this work with the NPC2 and envisage the scenario where, under certain conditions detected by the Environment Monitor, the Event Manager triggers a security policy update which permits packets with a certain label type to be processed. Another idea is to use the Event Manager itself to initiate the packet re-labelling process. This also has the result of permitting or denying processing (based on the label assigned). Both of these examples could be of use in a coalition environment where there may be a disparity between the types of information that can be shared, the label or context that has been applied and the security policy. Equally, the approach has potential application in other non-military contexts such as the futuristic brokerage scenario described in [77] which was described in earlier paragraphs of this chapter. In our view, the possibilities for application are really only limited by imagination.

There are numerous other environmental conditions that could be explored to provide policy trigger agents. Our scenario was based on elevating user capability in the event that a Commander was unavailable. Alternatives could include, but are not limited to, detected changes to the level of trust. In our scenario it is known that Domain AY is permitted to share with Domain B; however, it may not share with Domain C. Quite clearly there is a trust issue which raises the concern of information leakage between the two domains. There may be instances where it is necessary to place more trust in Domain C in which case simultaneous connectivity may be permitted. Certain events in the environment could be detected by autonomous sensors that change the way our agile mission group perceives its world, regards the level of trust in other members and subsequently the way it is required to operate. Our research scope has deliberately excluded trust because it is a vast and complex research area in its own right. However, it would certainly be an interesting extension to the work and one where we might opt to incorporate trusted platform technologies, such as those mentioned in Chapter 2, into the practical implementation of the NPC2.

Although there are very obvious opportunities to build more 'intelligence' into the Environment Monitor and Event Manager this is not something we have explored as part of this research. The whole philosophy behind our work is balancing flexibility with high assurance and our solution has therefore needed to operate at the more deterministic end of the flexibility scale. This approach should improve the acceptability of our ideas into a community which is necessarily conservative. There is however, a requirement to examine other cases for the NPC2 where it is permitted to adapt to the environment. In such a scenario it is postulated that the Environment Monitor and Event Manager would be provided with security goals and then allowed, within certain boundaries, to calculate the 'best' way of achieving those goals. By continuing to detect and utilise information from the environment the security policy could be updated to reflect changing needs that allow the goal to be maintained. Of course this presents a very large challenge for risk analysis and calculating the impact of the security model on the overall system and is one major reason why the approach was not explored for the NPC2. Our controlled flexibility allows the stakeholders to calculate impact at different states and refine the separation and sharing model accordingly. They can therefore have more confidence in the behaviour of the system when certain conditions are activated. A less deterministic model could quickly

become unmanageable, and we believe this would undermine the level of acceptability in a high assurance marketplace. Having said this, we do believe that a less deterministic NPC2 is a possibility for the future and certainly in non-military domains, particularly as research continues in fields devoted to investigating the security of enabling technologies such as Agent-based systems [52].

The research areas described in the preceding paragraphs can be categorised into different fields of interest. For example, extending the NPC2 within the SELinux platform, exploiting the technology into other markets and larger scenarios or adopting different technologies would clearly be of interest to industrial partners; probably as a series of industrial projects. By contrast, investigating the whole area of compromise paths including calculating growth, attaching probabilities to anticipated behaviours and assigning risk metrics would probably be better placed as either a purely academic project or perhaps a shared academic and industrial project. A number of academic institutions are actively researching the security of Agent-based technologies and continuing work could easily be conducted through a combination of academic and industry based research. Irrespective of who does the future work, we believe the subjects described are sufficiently interesting and useful to warrant further investigation.

This work commenced with a bold view that flexibility and high assurance were not usually considered as compatible partners because of the potential for unexpected and non-deterministic behaviour wreaking havoc in the level of confidence that users had in their own system and that of their communicating partners. However, it was also proposed that strict and inflexible separation did not provide a satisfactory answer since this impedes the capacity to share information and respond to changing requirements. Through our work with the NPC2 (in particular) and the NPD we believe that a compromise can be reached, which goes some way towards making the conjunction of flexibility and high assurance more palatable. Although there is still a lot of work to be done in this area, our research and the results documented throughout this thesis leads us to conclude that an important step has been taken in realising greater flexibility in security solutions within a demanding stakeholder community. It is concluded that our work moves us one stage further in achieving fit for purpose security.

References

[1] D. S. Alberts, J. J. Garstka, R. E. Hayes, and D.A. Signori, *Understanding Information Age Warfare,* Command and Control Research Program. (2001), [online] http://www.dodccrp.org/files/Alberts_UIAW.pdf. Last accessed 04/04/2008.

[2] The Open Group, Jericho Forum, 2007, [online] http://www.opengroup.org/jericho/Business_Case_for_DP_v1.0.pdf. Last accessed 04/04/2008.

[3] D. S. Alberts, *Information Age Transformation: Getting to a 21st Century Military,* Command and Control Research Program. (1996),[online] http://www.dodccrp.org/files/Alberts_IAT.pdf., revision 2002. Last accessed 04/04/2008.

[4] Ministry of Defence (MoD), JSP 777. Edition 1, (2004), [online] http://www.mod.uk/DefenceInternet/AboutDefence/CorporatePublications/SciencandTechnologyPublications/NEC/. Last accessed 02/04/2008.

[5] R. Ashby 1956, cited in *DCS-Wiki* [online] http://www.vs.uni-kassel.de/systems/index.php/Ashby_Theorems. Last accessed 14/5/08.

[6] D. Khulmann, and R. Gehring, "Trusted Platforms, DRM and Beyond" in E. Becker *et al.* eds. *Digital Rights Management; Technological, Economic, Legal and Political Aspects,* Berlin, Springer, (2003).

[7] R. L. Kay, "How to Implement Trusted Computing." Endpoint Technologies Associates, (2006), [online] https://www.trustedcomputinggroup.org/news/Industry_Data/Implementing_Trusted_Computing_RK.pdf. Last accessed 11/04/08.

[8] R. McMurran, F. McKinney, N.Tudor, and W. P. Millam, "Dependable System of Systems," Technical Paper, SAE International, (2006) [online] http://www.sae.org/technical/papers/2006-01-0597.

[9] D. Russell, and G.T.Gangemi Sr. *Computer Security Basics,* O'Reilly, (1991), p.106.

[10]CESG, "UK IT Security Evaluation and Certification Scheme," UK Scheme Publication No. 1, Issue 6.1, March 2006 [online] http://www.cesg.gov.uk/products_services/iacs/cc_and_itsec/media/formal-docs/uksp01.pdf. Last accessed 17/04/2008.

[11]CESG "Common Criteria: An Introduction," produced by Syntegra on behalf of the Common Criteria Implementation Board, [online] http://www.cesg.gov.uk/products_services/iacs/cc_and_itsec/media/intro-guides/criteria.pdf. Last accessed 17/04/08.

[12] CESG http://www.cesg.gov.uk/products_services/iacs/cc_and_itsec/media/intro-guides/cccert.pdf. Last accessed 17/04/08.

[13] CESG [online] http://www.cesg.gov.uk/about_us/index.shtml. Last accessed 17/03/08.

[14] CESG "Arrangement on the Recognition of Common Criteria Certificates in the Field of Information Technology Security", [online], May 2000, http://www.cesg.gov.uk/products_services/iacs/cc_and_itsec/media/formal-docs/ccra.pdf. Last accessed 17/04/08.

[15] Penguin Reference, *The Penguin Dictionary*, Penguin Books, UK, 2004, p. 729.

[16] QinetiQ, "Domain Based Security Architectures," Data Sheet, (2003) [online] http://www.qinetiq.com/home/security/digital_security/Domain_Based_Security.SupportingPar.0008.File.pdf. Last accessed 04/04/2008.

[17] C. Robinson, "Security Requirements Models to Support the Accreditation Process," presented at *2nd Annual Sunningdale Accreditors Conference*, UK, (2001). [online] http://www.qinetiq.com/home/security/digital_security/Domain_Based_Security.SupportingPar.0006.File.pdf. Last accessed 04/04/2008.

[18] K. Warrener, "Facilitating Risk Balance: an architectural approach," presented at *15th Annual Canadian Information Technology Security Symposium,* (2003) [online] http://www.qinetiq.com/home/security/digital_security/Domain_Based_Security.SupportingPar.0005.File.pdf

[19] K. Hughes, and S. Wiseman, "Analysis of information security risks: policy for protection through to implementation," in *Proceedings of 4th European Conference on Information Warfare and Security*, University of Glamorgan, UK, (2005).

[20] G. Chartrand, *Introductory Graph Theory*, Dover Publications, Inc. New York. (1977).

[21] D. R. Hill, and B. Kolman, *Modern Matrix Algebra* Prentice Hall, NJ. (2001).

[22] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P.Sommerlad, *Security Patterns: Integrating Security and Systems Engineering.* Wiley, UK. 2006.

[23] H. Janicke, and L. Finch, "The Role of Dynamic Security Policy in Military Scenarios", in *Proceedings of European Conference on Information Warfare,* Shrivenham, UK, 2007, pp.121-130.

[24] H. Janicke, and L. Finch, "The Role of Dynamic Security Policy in Military Scenarios", *Journal of Information Warfare*, Volume 6, Issue 3, pp.1-14. Mindsystems. (2007)

[25] L. Finch, and R. Vaughan, "Towards Fit for Purpose Security in Military Systems," *Proceedings of European Conference on Information Warfare*. Plymouth, UK. July 2008, pp. 71-80.

[26] C. Landwehr, "A Survey of Formal Models for Computer Security," *ACM Computing Surveys*, Volume 13, Issue 3, 1981. pp. 247-278.

[27] N. Daminaou, A. K. Bandara, M. Sloman, and E. C. Lupu, "A Survey of Policy Specification Approaches," (2002), [online] http://www.doc.ic.ac.uk/~mss/Papers/PolicySurvey.pdf. 2002. Last accessed 07/04.2008.

[28] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley. (2001).

[29] D. Clark, and D. Wilson, "A Comparison of Commercial and Military Computer Security Policies," in *Proceedings of the 1987 IEEE Symposium on Security and Privacy,* (1987), pp. 184 -194.

[30] J. Anderson, "Computer Security Technology Planning Study". ESD-TR-73-5, Volume II, (1972). [online] http://seclab.cs.ucdavis.edu/projects/history/papers/ande72.pdf. Last accessed 17/04/08.

[31] J. M. Rushby, and B. Randall, "A Distributed Secure System," in *proceedings of 1983 IEEE Symposium on Security and Privacy*, (1983), p.127.

[32] J. Alves-Foss, W. S Harrison, P. Oman, and C. Taylor, "The MILS Architecture for High Assurance Embedded Systems," *International Journal of Embedded Systems,* Volume 2, No 3/4, pp. 239-247, (2006).

[33] J.M. Rushby, "Design and Verification of Secure Systems," *ACM SIGOPS Operating Systems Review*, Volume 15, Issue 5, pp. 12-21, (1981)

[34] R. J. DeLong, "Separation Kernel for Secure Real-Time Operating System", originally published in *Boards and Solutions Magazine*, Feb 2008. [online] http://www.embedded-control-europe.com/c_ece_knowhow/163/BASfeb08p22.pdf.

[35] C. Howard, "Green Hills Software Introduces Secure Networking Platform," published in *Military & Aerospace Electronics,* (2007), [online] http://www.ghs.com/articles/GHS_70310_156e_MAE_01_07.pdf. 2007. Last accessed 07/04/2008

[36] D. Bell and L. LaPadula, "Secure Computer System unified exposition and Multics interpretation", Technical Report MTR-2997, MITRE, (1975)

[37] K. J. Biba, "Integrity Considerations for Secure Computer Systems", MTR-3153, The Mitre Corporation, April 1977.

[38] D.F.C Brewer and N.J. Nash, "The Chinese Wall Security Policy", in *IEEE Symposium on Security and Privacy*, (1989).

[39] Bishop, M, *Computer Security: Art and Science,* Pearson Education Inc, Addison-Wesley, Boston, (2003).

[40] Z. Hayat, J. Reeve, and C. Boutle, "Domain Based Security: Improving Practices," in *First Annul Conference on Domain Based Security*, QinetiQ, Malvern, (2005),[online] http://eprints.ecs.soton.ac.uk/12276/. Last accessed 08/04/2008

[41] A. Jones and D. Ashenden, *Risk Management for Computer Security*. Elsevier, (2005).

[42] Insight Consulting [online] http://www.insight.co.uk/products/cramm.htm. Last accessed 07/05/08.

[43] MathWorks [online] http://www.mathworks.com/products/matlab/description1.html last accessed 07/04/2008

[44] AS/NZS Australian Standard Risk Management, [online] http://www.wales.nhs.uk/ihc/documents/A.4.1.4_Australia_and_New_Zealand_Methodology_AS_NZ%204360_1999.pdf. (1999), pp. 41-42. Last accessed 07/04/2008.

[45] V. K. Balakrishnan, *Graph Theory*. McGraw-Hill. (1997), p1.

[46] R. Metcalfe, cited in G. Gilder, "Metcalf's Law and Legacy", Forbes, (1993) [online] http://www.seas.upenn.edu/~gaj1/metgg.html. Last accessed 08/04/2008.

[47] B. Briscoe, A. Odlyzko, and B. Tilly, "Metcalfe's Law is Wrong" *IEEE Spectrum,* (2006),[online] http://spectrum.ieee.org/print/4109. Last accessed 08/04/2008.

[48] A. Cau, F. Siewe, and H. Zedan, "A Compositional Framework for Access Control Policies Enforcement", in *Proceedings of the 2003 ACM Workshop on Formal Methods in Security Engineering*, 2003, pp. 32-42.

[49] H. Janicke, A. Cau, F. Siewe, H. Zedan, and K. Jones, "A Compositional Event & Time-based Policy Model" in *Proceedings of 2006 IEEE Workshop on Policies for Distributed Systems and Networks*, London, Ontario, Canada, (2006), pp. 173-182.

[50] H. Janicke, "The Development of Secure Multi-Agent Systems", PhD Dissertation, STRL, DeMontfort University, Leicester (UK), (2007).

[51] A. Cau, B. Moszkowski, and H. Zedan, "The ITL homepage" (2007), [online], DMU, http://www.cse.dmu.ac.uk/STRL/ITL.

[52] H. Janicke, F. Siewe, K. Jones, A. Cau, and H. Zedan, "Analysis and Runtime-Verification of Dynamic Security Policies", in *Proceedings of DAMAS 2005* at AAMAS'05, Utrecht, Netherlands (2005).

[53] D. Baker, "Fortress Built upon Sand," in *Proceedings of the New Security Paradigms Workshop,* Calafornia, US. (1996), pp.148-153.

[54] P. A. Loscocco, P. A. Smalley, D. A. Muckelbauer, R. C.Taylor, J. S. Turner, and J.F. Farrell, "The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments," in *Proceedings of National Information Systems Security, Security Conference*, (1998), pp. 303-314.

[55] National Security Agency. SELinux general overview [online] http://www.nsa.gov/selinux/index.cfm. Last accessed 08/04/2008.

[56] W. E. Boebert, and R. Y. Kain, "A Practical Alternative to Hierarchical Integrity Policies," in *8th National Computer Security Conference* (1985) cited in Anderson, R. Security Engineering: A Guide to Building Dependable Distributed Systems, p145.

[57] F. Mayer, K. MacMillan, and D. Caplan, *SELinux by Example.* Prentice Hall, (2007).

[58] C. J. PeBenito, F. Mayer, and K. MacMillan, "Reference policy for Security Enhanced Linux" presented at Second Security Enhanced Linux Symposium, Baltimore, MA, (2006), http://www.tresys.com/files/docs/RefPol-SELinux-Symp.pdf.

[59] J. Athey, C. Ashworth, F. Mayer, and D. Miner, "Towards Intuitive Tools for Managing SELinux: Hiding the Details but Retaining the Power," presented at Third Security Enhanced Linux Symposium, Baltimore, MA, (2007), [online] http://selinux-symposium.org/2007/papers/07-brickwall.pdf. Last accessed 08/04/2008.

[60] Tresys [online] http://www.tresys.com/files/docs/cond-readme.txt.

[61] M. Gregory, and P. Loscocco, "Using the Flask Security Architecture to Facilitate Risk Adaptable Access Controls", presented at Third Security Enhanced Linux Symposium, Baltimore, MA, (2007) [online] http://selinux-symposium.org/2007/papers/03-RAdAC.pdf. Last accessed 08/04/2008.

[62] B. Pollett, M. Butler, and J. Hale, "Dynamic Policy Enforcement in a Network Environment", Second Security Enhanced Linux Symposium, Baltimore, MA, (2006), [online] http://selinux-symposium.org/2006/papers/06-dynamic-enforcement.pdf. Last accessed 08/04/2008.

[63] H. Von Moltke, *Moltke on the Art of War,* Selected Writings edited by D. Hughes, The Random House Ballentine Publishing Group, 2003, p. 92.

[64] QinetiQ, *Planning and Documenting InfoSec Course*, training course materials, QinetiQ, 2005.

[65] J. Saltzer, and M. Schroeder, "The Protection of Information in Computing Systems," in Proceedings IEEE. Volume 63, Issue 9, (1975), pp.1278-1308.

[66] C. Perkins, E. Belding-Royer, and S. Das, "Ad-Hoc On Demand Distance Vector Routing", RFC 3561, (2003) [online] http://www.apps.ietf.org/rfc/rfc3561.html. Last accessed 10/04/08.

[67] Tresys [online] http://oss.tresys.com/projects/refpolicy. Last accessed 10/04/08.

[68] Tresys [online] http://oss.tresys.com/projects/setools/wiki/seaudithelp. Last accessed 10/04/08.

[69] Tresys Open Source Projects SELinux Integrated Development Environment (SLIDE)Introduction [online] http://oss.tresys.com/projects/slide. Last accessed 10/04/08.

[70] Eclipse Community overview of Eclipse integrated development environment [online] http://www.eclipse.org/. Last accessed 10/04/08.

[71] Tresys Apol Help files [online] http://oss.tresys.com/projects/setools/wiki/helpFiles. Last accessed 10/04/08.

[72] Tresys [online] http://oss.tresys.com/projects/setools. Last accessed 10/04/08.

[73] S. Shimko, and J. Brindle, "Securing Inter-process Communications in SELinux," presented at Third Security Enhanced Linux Symposium, Baltimore, MA, (2007) [online] http://www.tresys.com/files/docs/Securing-Inter-process-Communications-in-SELinux.pdf.

[74] HMG InfoSec Standard 1, Part One, Technical Risk Assessment. Issue 3.1, July 2007. Not Protectively Marked.

[75] D. Ferraiolo, and D. Kuhn, "Role-Based Access Control" in *Proceedings of the NIST-NSA National (USA) Computer Security Conference*, (1992), pp. 554-563.

[76] Journal of Information Warfare, Mindsystems. [homepage] http://www.mindsystems.com.au/services/jiw/

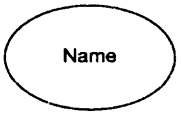
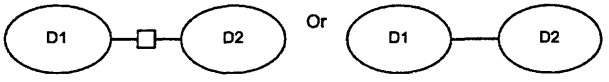
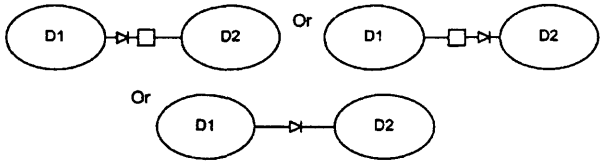
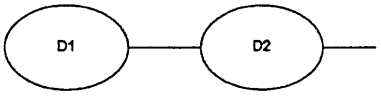
[77] P. Chen Cheng, P. Rohatgi, C. Kesser, P.A. Karger, G. M. Wagner and A. Schuett Reninger, "Fuzzy Multi-Level Security: An Experiment on Quantified Risk-Adaptive Access Control" in *Proceedings of the IEEE Symposium on Security and Privacy, (2007)*, pp.222-230.

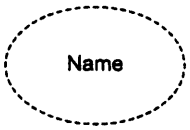
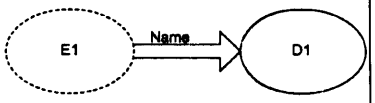
[78] T. Jaeger, D. King, K. R. Butler, S. Hallyn, J. Latten and X Zhang, "Leveraging IPSec for Mandatory Per-Packet Access Control", IEEE Securecomm and Workshops, IEEE Xplore, (2006), pp. 1-9.

Appendix A – DBSy Modelling Notation

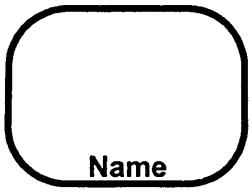
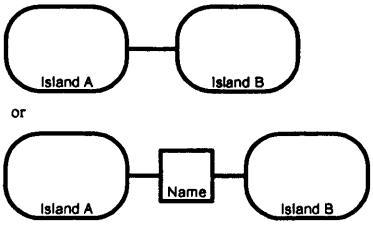
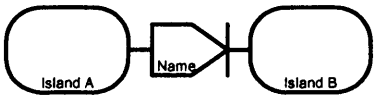
The tables in this section have been extracted from the Domain Based Security (DBSy) User Guide No 3: InfoSec Models Quick Reference Guide. This guide has been made available by kind permission of QinetiQ who have agreed to allow the author to copy and include the following symbols and their descriptions used in the production of the thesis. The tables which follow are by no means the complete list of symbols in the quick reference guide.

Infosec Business Models

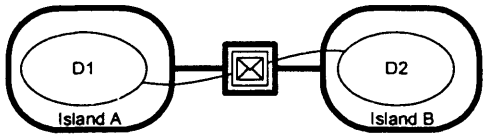
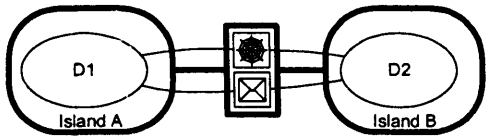
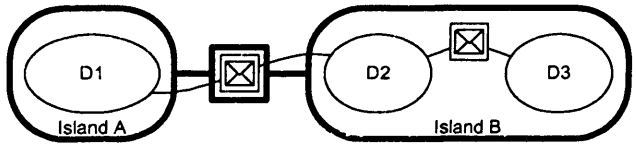
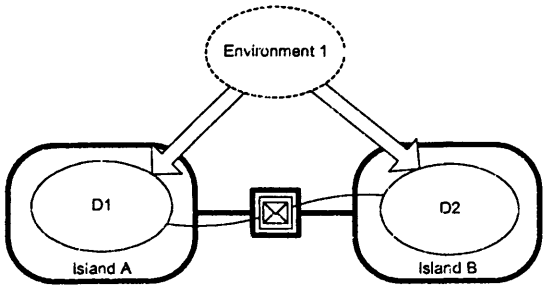
Domain	A logical set of facilities where software processes information on behalf of those people who are members of the domain. Each **domain** has a name that is unique within the model. A domain is characterised by the minimum security requirements for its members and maximum security properties of data handled within it. People who work in a domain may share information with relative freedom.	
Internal Domain		Any **domain** that is to be implemented in whole or in part by a project or that is otherwise deemed to be part of the 'focus of interest' for the purpose of the model.
External Domain		Any **domain** for which a project has no responsibility or that is considered to be outside the 'focus of interest' for the purpose of the model. (N.B. Onward connections may exist)
Connection	Enables members of a **domain** to share information in some way with members of another **domain**. Two **domains** are connected if information may be transferred from one **domain** to the other.	
Two-way connection (unspecified type)		A two-way **connection** permits business information transfer in both directions.
One-way connection (unspecified type)		A one-way **connection** permits business information transfer in one direction only, as indicated by the diode (in this case only D1 to D2).
Onward Connections		D2 has further unspecified business connections.

Environment		The physical place where people work and where equipment and media are located. Each **environment** has a name that is unique within the model.
Portal		Enables **domain** members in a specific **environment** to interact with software acting on their behalf in a **domain**. The name is optional.
Connection types	Different types of **connection** permit people to share information in different ways. An icon inside the square **connection** symbol indicates the type of **connection**. A name may be also be used to specify the type more precisely.	
Message connection		Enables a person to send information to specifically nominated people.
Shared Data Repositories	Enables information created by members of one domain to be published so that members of another domain may observe it. Several types of repository are defined.	
File store		Enables people to publish and request named files in a hierarchical directory structure.

Infosec Infrastructure Models

An infrastructure model defines the "impenetrable" boundaries of a system. An impenetrable boundary is so strong that people outside it can be ignored in the risk assessment. For protection of protectively marked HMG information this means in practice either: a) Physical separation (with necessary tempest protection) b) Cryptography of suitable grade.		
Island		An isolated island represents a computer system that is separated from all other computer systems by an impenetrable boundary. An island has a name that is unique within the model.
Causeways	Causeways provide an identifiable and manageable point of connection between two or more islands. They allow sharing of information between the islands via non by-passable controls.	
Two-way causeway		An impenetrable boundary separates the two islands from all other computer systems and ensures that the interaction between the islands is implemented by the causeway and by no other means. The causeway name is optional.
One-way causeway		Data transfer is in one direction only, with no physical means to transfer an electronic signal in the opposite direction.

Infosec Architecture Models

Infosec Architecture models show how the business requirements shown in an Infosec Business model are implemented within a security architecture depicted by an Infosec Architecture model. Curved lines are used for connecting business model components to make them distinct from infrastructure connections.	
	Each island supports a single domain. If an island supports only one domain its name may be omitted from the diagram.
	Causeway implements two connections
	Island supports more than one domain.
	Members of environment 1 can access information in D2 and D1 through the portal connections provided to the island infrastructures.

Advanced Model Notation

Refinement	*High level model*	*Example refinement*
Refinement of a single domain (B) to show two components domains, BX & BY. In this case only BX members can email domain A.