COPYRIGHT AND CITATION CONSIDERATIONS FOR THIS THESIS/ DISSERTATION

- Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

- NonCommercial — You may not use the material for commercial purposes.

- ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

How to cite this thesis

A MODEL FOR THE EVALUATION OF CONTROL WITH REFERENCE TO A SIMPLE PATH CONTEXT MODEL IN A UNIX ENVIRONMENT

G. S. DU PLESSIS

UNIVERSITY OF JOHANNESBURG
A model for the evaluation of control with reference to a simple path context model in a UNIX environment

By

Gerrit Steyn Du Plessis

SHORT DISSERTATION

Submitted for the partial fulfilment of the requirements for the degree

MASTER IN COMMERCE IN COMPUTER AUDITING IN THE FACULTY OF ECONOMIC AND MANAGEMENT SCIENCES AT THE RAND AFRIKAANS UNIVERSITY

STUDY LEADER: PROF. A. DU TOIT

JOHANNESBURG

NOVEMBER 1995
DECLARATION

I declare that this research essay hereby submitted to the Rand Afrikaans University for the degree of Master of Commerce, except to the extent acknowledged in the text, is my own unaided work and has not been submitted previously for any degree to any other university.

GERRIT STEYN DU PLESSIS
## INDEX

### CONTENTS

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td>i</td>
</tr>
<tr>
<td>OPSOMMING IN AFRIKAANS</td>
<td>iv</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>ESTABLISHING THE FRAMEWORK</td>
<td>9</td>
</tr>
<tr>
<td>APPLICATION OF THE MODEL IN THE UNIX ENVIRONMENT</td>
<td>27</td>
</tr>
<tr>
<td>DEVELOPMENT OF AN AUDIT PROGRAMME</td>
<td>57</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>72</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>76</td>
</tr>
</tbody>
</table>
SYNOPSIS

1. PROBLEM DESCRIPTION

Information and the IT systems that support it are important business assets. Their availability, integrity and confidentiality are essential to maintain an organisation's competitive edge, cash flow, profitability, company image and compliance with legal requirements.

Organisations world-wide are now facing increased security threats from a wide range of sources. Information systems may be the target of a range of serious threats including computer-based fraud, espionage, sabotage, vandalism and other sources of failure or disaster.

New and more sophisticated sources of damage continue to emerge, such as threats of viruses and computer hackers. Information security threats are expected to become more widespread, more ambitious and increasingly sophisticated.

At the same time organisations could become more vulnerable to security threats, from greater dependence on IT systems and services. The shift towards distributed computing and the growth of networking reduces the scope for central, specialist control of IT facilities and present new opportunities for unauthorised access to computer systems. The need for improved computer security to ensure the availability, integrity and confidentiality of information systems is therefore of paramount importance.

UNIX's reputation as a non-secure operating system comes from practice rather than from theoretical design. In the words of one of UNIX's designers:

"It was not designed from the start to be secure. It was designed with the necessary characteristics to make security serviceable" (Ritchie, 1989; as quoted by Garfinkel & Spafford, 1993:8).

It is therefore necessary to develop alternative control models to make security in UNIX systems serviceable. Studies by Boshoff (1985, 1990) led to the development of the Access Model and the Path Context Model as methods for evaluating security issues in complex computer systems. The credibility of these two models has been widely accepted as can be seen in the publication of findings in various international journals.

In the light of this, the main reasons for this research essay can be described as follow:

- Applicability of the Access Model (Boshoff, 1985) and the Path Context Model (Boshoff, 1990) to the UNIX environment;
- Recognition of the risks associated with multiple layers of system software which are present in complex computer systems; and
- The misconception that exists with many users that UNIX is a non-secure system.
2. PURPOSE AND AIM OF THIS RESEARCH ESSAY

The main purpose of this research essay is to provide an overall evaluation of the key components of the selected environment from the standpoint of controls and security by developing an audit programme or checklist that addresses the control and security issues pertaining to the key components of the UNIX operating system.

3. METHODOLOGY

On the basis that security in complex computer systems is of great importance and in the light of perceived UNIX security features there is a need to develop a framework for the evaluation of controls in the predefined UNIX environment. The need for computer security has been widely accepted and shall therefore not be discussed in this essay.

The applicability of the Access Path Model and the Path context model has been proved (Boshoff 1985, 1990). These models are applied to the predefined UNIX environment. The environment will be defined in terms of this model. A description of the overall framework and key components in the framework will be carried out.

The framework will then be expanded and details of key components that are important for satisfying the predefined control objectives documented.

An audit programme that will detail important system software components and procedures to be followed for each component to satisfy the control objectives will be developed.

Conclusions will be made on whether the Access Path Model and the Path Context Model (Boshoff 1985, 1990) can be used to evaluate whether the installation will achieve the control objectives set out and consequently whether there is adequate security or security features in the predefined UNIX environment.

The methodology followed in this research is set out in Figure 1 on page iii.

4. Summary of Results

By applying the Access Path and Path Context Models (Boshoff 1985, 1990) to the predefined UNIX environment a model was developed for the evaluation of the control facilities within the different layers of the UNIX operating system. This allowed successful evaluation and analysis of control facilities and audit issues in the predefined environment.

The analysis revealed that each layer of the UNIX operating system as defined in the model can act as a screen preventing the users from getting to the next layer.
Using the methodology established in this research essay, and the Access Path and Path Context Models, it was possible to evaluate the predefined UNIX environment and develop an audit programme for the review of a UNIX installation.

It was found that in each of the different layers, except for the networking layer of the predefined path there was sufficient controls to ensure that:

- access overall are restricted to system objects on a per user or per group basis, as well as controls to limit the propagation of access rights;
- users are only allowed on networks after they have uniquely identified themselves;
- An audit trail of all access of system objects are maintained.

Because of the impact of the security weaknesses of the network layer on the rest of controls in the system, it was found that the operating system have more risk factors than controls to control these risks. It is therefore concluded that all control objectives will not be met by the UNIX operating system without the use of an external computer security package to compliment the controls of the UNIX operating system.

Because each layer in the model consists of different components, all of which do not need to be active for the UNIX operating system to function properly, the auditor must consider whether the components active on the installation under review are required and the effect thereof on the overall computer security, as some of these components present the installation with risk factors that cannot be effectively and efficiently countered by any of the layers in the model.

Figure 1: Research Essay Methodology
‘n Model vir die evaluering van kontrolemaatreëls
in ‘n eenvoudige roete konteks model in die
UNIX omgewing

deur

Gerrit Steyn Du Plessis

OPSOMMING VAN ‘N VERHANDELING INGEDIEN VIR DIE GRAAD
MAGISTER COMERCI IN REKENAROUDITERING IN DIE
FAKULTEIT EKONOMIESE - EN BESTUURSWETENSKAPPE AAN
DIE RANDSE AFRIKAANSE UNIVERSITEIT

STUDIELEIER: PROF. A. DU TOIT

JOHANNESBURG

NOVEMBER 1995
OPSOMMING IN AFRIKAANS

1. INLEIDING

Die doel van die opsomming is om die agtergrond, metodiek en gevolgtrekkings van die navorsing uitgevoer op die geselekteerde omgewing weer te gee. Die opsomming sal onder die volgende hoofde uiteengesit word:

• Agtergrond, probleemomskrywing en doel van die navorsing;
• Metodiek gevolg in die navorsing; en
• Gevolgtrekkings.

2. AGTERGROND, PROBLEEMOMSKRYWING EN DOEL VAN DIE NAVORSING

Met die oog op internasionale mededinging wat al strawwer word tussen ondernemings, is hedendaagse ondernemings genoodsaak om te herstruktureer sodat bedrywighede meer funksioneel en effektief uitgevoer kan word. Dit beteken dat ondernemings ook die voordelige wat nuwe en verbeterde inligtingstelsels bied sal moet benut en bestuur. Die eienskappe van nuwe en verbeterde inligtingstelsels het 'n wesenlike impak op bestuur van die onderneming en beheer structure binne die onderneming.

Die toename in internasionale kompetisie veroorsaak dat ondernemings toenemende vertroue op inligtingstelsels plaas as deel van die totale besigheid strategie. Die automatisering en rekenarisering van organisatoriese funksies het tot gevolg dat meer gesofistikeerde beheer meganismes in rekenaarstelsels ingebou word. Dit het tot gevolg dat die tegnologie van beheer- en kontrole maatreëls hoofsaaklik as volg verander:

• Gebruikers- en nie gesofistikeerde rekenaar kontroles van vorige generasie rekenaar stelsels, word nou in hardware en sagteware ingebou; en
• Beheer- en kontrole maatreëls wat voorheen nie moontlik was om in te stel en toe te pas nie, word nou moontlik gemaak deur die ontwikkeling van nuwe generasie hardeware, stelselsagteware en toepassingsprogrammatuur.
Tradisionele kontroles wat normaalweg geneig was om, “om die rekenaar” te wees, is dus in moderne inligtingstelsels nie meer effektief nie en in sommige gevalle heeltemal oorbedig.

Alternatiewe beheer en kontrole modelle, wat die risikos van moderne inligtingstelsels aanspreek, is dus nodig om die tegnologie van kontroles in hierdie inligtingstelsels te kan begryp sodat die kontrole maatreëls wat deur hierdie inligtingstelsels verskaf word behoorlik geimplimenteer kan word.

Studies deur Boshoff (1985;1990) het gelei tot die ontwikkeling van die toegangsroete Model en die Roete Konteks Model as geldige hulpmiddel vir die evaluerings van beheer en kontrole maatreëls in komplekse rekenaarstelsels. Die bevindinge van Boshoff is wyd gepubliseer in invloedryke internasionale publikasies wat verdere steun verleen aan die toepaslikheid en aanvaarding van hierdie modelle.

Die doel van hierdie navorsing is om te bepaal of die modelle wat deur Boshoff daargestel is vir die evaluerings van rekenaar sekuriteit, op die UNIX omgewing toepas kan word om te bepaal of die UNIX stelselsagteware voldoende kontroles bevat om aan erkende oudit doelwitte te voldoen. Met behulp van die modelle deur Boshoff daargestel sal 'n ouditprogram ontwikkel word om die verskillende komponente van die UNIX stelsel aan die hand van erkende oudit doelwitte te evalueer.

2. METODIEK

Die metodiek gevolg in hierdie navorsing word in Figuur 1 saamgevat. Die rasionaal vir die benadering gevolg kan as volg verduidelik word:

- Die behoefte aan rekenaar sekuriteit in die huidige kompetereende internasionale besigheidsomgewing word aanvaar. Die behoefte aan rekenaar sekuriteit is ook deur die studies van Boshoff (1985;1990) bevestig.
- Die tradisionele benadering wat normaalweg gevolg is by die evaluerings van rekenaarstelsels is steeds effektief, toepaslik en geldig in die minder komplekse rekenaaromgewing.
Hoofstuk 2

Hoofstuk 3
Identifiseer en analyseer die komponente wat 'n effek op kontroedoelwitte het.

Hoofstuk 4
Ontwikkel 'n auditprogram vir die geïdentificeerde komponente aan die hand van kontroedoelwitte.

Hoofstuk 5
Maak 'n gevolgtrekking oor rekenaars sekuriteit

FIGUUR 1: Metodologie
Die Toegangsroete Model ontwikkel deur Boshoff is toegepas op 'n geïdentifiseerde omgewing naamlik die UNIX omgewing. Die omgewing sal geïdentifiseer word in terme van die model en sal diagrammaties voorgestel word. 'n Konsepsuele beskrywing van die sleutel komponente binne die raamwerk sal gegee word.

Die raamwerk sal dan gebruik word vir die opstel van 'n auditprogram. Die auditprogram sal die verskillende komponente sowel as die eienskappe eie aan die komponente aan die hand van die kontrole doelwitte evalueer.

Die gevolgtrekkings wat gemaak word, sal reflekteer of die Toegangsroete en Konteks Model (Boshoff 1985;1990) gebruik kan word om sekuriteit in die geïdentifiseerde UNIX omgewing te evalueer, deur middel van die ontwikkeling van 'n auditprogram, wat die verskillende komponente soos geïdentifiseer aan die hand van kontrole doelstellings evalueer. Op grond van laasgenoemde bevindings sal gevolgtrekkings gemaak word, wat sal reflekteer of die kontroledoelstellings bereik is en of daar gevolglik doeltreffende rekenaar sekuriteit in die voorafbepaalde omgewing is.

3. GEVOLG TREKKING

Deur middel van die toepassing van die Toegangsroete en Konteks Model (Boshoff 1985;1990) is die verskillende lae en komponente van die UNIX bedryfstelsel sowel as die eienskappe van die komponente wat 'n effek op rekenaar sekuriteit het geïdentifiseer.

Hierdie toepassing van die model het aan die lig gebring dat die bedryfstelsel in verskillende lae, elk met sy eie komponente, opgedeel kan word. 'n Gebruiker van die stelsel moet deur die verskillende lae beweeg voordat hy/sy toegang tot data of programmatuur in die stelsel verkry. Elke laag tree op as 'n weerstandsvlak wat die gebruiker verhoed om toegang tot die volgende vlak te verkry.

Die model wat ontwikkel is vir die evaluering van die verskillende lae en hul onderskeie komponente deur middel van die opstel van 'n auditprogram wat die komponente evalueer aan die hand van kontrole doelwitte word in Figuur 2 uiteengesit.
Deur 'n ouditprogram te ontwikkel wat aan die hand van die kontroledoelwitte daargestel, die verskillende lae en hul komponente evalueer kan die ouditeur evalueer of die UNIX installasie onder oudit wel die gestelde kontroledoelwitte bereik.

Die gevolgtrekking waartoe gekom is, is dat sekere komponente in die verskillende lae binne die UNIX omgewing soos bv. rsh nie aan die kontroledoelwitte voldoen nie terwyl ander wel aan die kontroledoelwitte voldoen. Aangesien die UNIX bedryfstelsel
modulêr is en al die komponente nie in 'n gegewe omgewing inwerking hoef te wees vir die bedryfsteel om te kan funksioneer nie, sal die ouderiteur dit moet oorweeg of die komponent wel in werking moet wees al dan nie, en wat die effek hiervan op die kontroledoelwitte is.
INTRODUCTION

1.1 BACKGROUND

As global competition in the twenty first century looms closer on the horizon, organisations are restructuring to streamline operations and simultaneously take advantage of the advances in information systems technologies. These changes have an impact on the ways that business and government entities operate. They are having profound implications for the management and operational control structures within organisations world-wide.

The emphasis on competition implies an ever increasing reliance on computing as a major component in the downsizing strategies for many organisations. Computerising organisational functions is dictating the incorporation of more powerful control mechanisms into computers and networks, both hardware-based and software-based. The fundamental characteristics of these controls are evolving at the same rate and in the same manner as the underlying computing and networking technologies are evolving.

The technology of controls is changing in two ways:

- Manual controls and unsophisticated computer controls from previous generations of technology are being imbedded in the hardware and software of information systems.
- New control techniques that were not possible before are being enabled by the development of entirely new generations of computer hardware, systems software, and applications software.

Traditional controls that tended to be ‘around’ the computer have therefore become inefficient and in some cases even redundant in modern information systems.

Alternative control models that address risks being faced by organisations with modern information systems are therefore necessary to understand the technology of controls in these modern information systems that limit the risks faced by organisations.

1.2 UNIX HISTORY - A BRIEF OVERVIEW

In the mid 1960's, AT&T and Bell Laboratories were participating in an effort to develop a new operating system called MULTICS. MULTICS was intended to supply large scale computing services as a utility (Armstrong, Douba, Lee Henry, Rose, Rummel, Parker, Marshall, Dippold, Negus, Valley, Smith, Taylor, Weinstein, Till, 1994:8)
In 1969, Bell Laboratories pulled out of the MULTICS effort, and the members of the Computing Science Research centre were left with no computing environment. Ken Thompson and Dennis Ritchie developed and simulated an initial design for a file system that later developed into the UNIX file system. At this stage the system was developed to take advantage of a PDP-7 computer (Armstrong, Douba, et al. 1994:8).

An early project that helped UNIX to its success was its deployment to do text processing for the patent department at AT&T. This project moved UNIX to a PDP-11 computer and resulted in the system being known for its small size. Shortly after this the C programming language was developed on and for UNIX, and the UNIX operating system itself was rewritten in C (Armstrong, Douba, et al. 1994:8).

AT&T was not allowed by law to market the computer system, so it had no way of selling this creative work form Bell Labs. The popularity of UNIX however grew through internal use at AT&T and licensing to universities for educational use. In 1977 commercial licenses were granted, and the first UNIX vendor, Interactive Systems Corporation, began selling UNIX systems for office automation (Armstrong, Douba, et al. 1994:8).

Later UNIX versions developed at AT&T included System III and several releases of System V. The two most recent releases are System V release 3 (SVR 3.2) and Release 4 (SVR 4.2)

1.3 PROBLEM DESCRIPTION

Information and the IT systems that support it are important business assets. Their availability, integrity and confidentiality are essential to maintain an organisation's competitive edge, cash flow, profitability, company image and compliance with legal requirements.

Organisations world-wide are now facing increased security threats from a wide range of sources. Information systems may be the target of a range of serious threats including computer-based fraud, espionage, sabotage, vandalism and other sources of failure or disaster.

New and more sophisticated sources of damage continue to emerge, such as threats of viruses and computer hackers. Information security threats are expected to become more widespread, more ambitious and increasingly sophisticated.

At the same time organisations could become more vulnerable to security threats, from greater dependence on IT systems and services. The shift towards distributed computing and the growth of networking reduces the scope for central, specialist control of IT facilities and present new opportunities for unauthorised access to computer systems. The need for improved computer security to ensure the availability, integrity and confidentiality of information systems is therefore of paramount importance.
UNIX's reputation as a non-secure operating system comes from practice rather than from theoretical design. In the words of one of UNIX's designers:

"It was not designed from the start to be secure. It was designed with the necessary characteristics to make security serviceable" (Ritchie, 1989; as quoted by Garfinkel & Spafford, 1993:8).

It is therefore necessary to develop alternative control models to make security in UNIX systems serviceable. Studies by Boshoff (1985,1990) led to the development of the Access Model and the Path Context Model as methods for evaluating security issues in complex computer systems. The credibility of these two models has been widely accepted as can be seen in the publication of findings in various international journals.

In the light of this, the main reasons for this research essay can be described as follow:

- Applicability of the Access Model (Boshoff, 1985) and the Path Context Model (Boshoff, 1990) to the UNIX environment;
- Recognition of the risks associated with multiple layers of system software which are present in complex computer systems; and
- The misconception that exists with many users that UNIX is a non-secure system.

1.4 PURPOSE AND AIM OF THIS RESEARCH ESSAY

The main purpose of this research essay is to provide an overall evaluation of the key components of the selected environment from the standpoint of controls and security by developing an audit programme or checklist that addresses the control and security issues pertaining to the key components of the UNIX operating system.

1.5 SCOPE AND LIMITATIONS

1.5.1 Predefined environment

An unfortunate side effect of UNIX's popularity is that there are many different versions of UNIX available on the market. Many UNIX vendors have also modified the basic behaviour of some of their system commands and system features. The Access Path Model (Boshoff, 1985) will be used to develop a control model for the standard version of UNIX System V Release 3.2 and Release 4.2. Standard and important utilities will be incorporated in the control model.
1.5.2 User and general controls

Due to the sophisticated nature and the complexity of the predefined environment the extensive use of user controls are both inefficient and inappropriate in that these controls seem to be impractical.

In this essay, controls that exists within system software will be explored and user controls will only be investigated as compensating controls where serious weaknesses in controls contained within system software exist.

The concepts of user and general controls have been well documented by the institutes governing auditing standards, such as the American Institute of Certified Public Accountants (AICPA), the South African and Canadian Institute of Chartered Accountants (SAICA, CICA).

1.5.3 Control Objectives

The control model being developed is driven by a number of control objectives whose validity and appropriateness have been proven by publication thereof by the various institutes governing auditing standards such as SAICA, CICA and AICPA. These control objectives are:

- **Completeness** of input, processing, updating of files and output;
- **Accuracy** of input, processing, updating of files and output;
- **Validity** of processing;
- **Integrity** of data both in transient and static state; and
- **Continuity** - ensuring that the product is operating, and is capable of operating in accordance with business practice and management expectations.

The validity and appropriateness of these control objectives as well as the Access Model are assumed based on the published research by Boshoff (1985, 1990).

1.5.4 Limitations and Exclusions

This essay is merely the application of a model to a specified environment. Certain exclusions apply to this research essay as follows:

- Non-standard UNIX utilities and versions are excluded as it will be impossible to evaluate all the UNIX utilities available on the market;
- Security packages are ignored, as they fall outside the objectives of this essay which is to evaluate the ability of the specified system software to meet the control objectives;
- Specific versions of the system software have been dealt with as many different versions are available due to the history of UNIX; and
- Issues not having a direct impact on the system have been ignored.
1.6 METHODOLOGY

On the basis that security in complex computer systems is of great importance and in the light of perceived UNIX security features there is a need to develop a framework for the evaluation of controls in the predefined UNIX environment. The need for computer security has been widely accepted and shall therefore not be discussed in this essay.

The applicability of the Access Path Model and the Path context model has been proved (Boshoff 1985, 1990). These models are applied to the predefined UNIX environment. The environment will be defined in terms of this model. A description of the overall framework and key components in the framework will be carried out.

The framework will then be expanded and details of key components that are important for satisfying the predefined control objectives documented.

An audit programme that will detail important system software components and procedures to be followed for each component to satisfy the control objectives will be developed.

Conclusions will be made on whether the Access Path Model and the Path Context Model (Boshoff 1985, 1990) can be used to evaluate whether the installation will achieve the control objectives set out and consequently whether there is adequate security or security features in the predefined UNIX environment.

The methodology followed in this research can be set out in Figure 1.1.

![Figure 1.1: Research Essay Methodology](image-url)
1.7 Literature Survey

To carry out an extensive literature survey relative to the topic under consideration will serve no meaning and useful purpose because:

- This essay relates to a predefined environment that has not been covered in its entirety before; and
- To restate the difference between general and user controls, and to provide definitions for each will serve no meaningful purpose as this topic has been extensively dealt with in various texts including Stettler (1977:524), AICPA (1984:6; 1977:25-62), Boshoff (1985:15) and CICA (1973:2).

Instead the previously published work of Boshoff (1985:1990) and the work carried out by the author will form the basis for this research essay.

The control objectives chosen are those suggested by Boshoff (1985, 1990). Proof of the validity of these control objectives are documented in Boshoff’s research essays (1985,1990) and is assumed correct for this research essay.

Support is also given to these control objectives by various institutes governing auditing standards such as the AICPA (1984:6; 1988:10).

The proof of the validity of the Access Path Model and the path Context Model are assumed (Boshoff 1985,1990). The publication of Boshoff’s work in various international journals adds weight to the acceptance of these models.

1.8 Summary of Results

By applying the Access Path and Path Context Models (Boshoff 1985, 1990) to the predefined UNIX environment a model was developed for the evaluation of the control facilities within the different layers of the UNIX operating system. This allowed successful evaluation and analysis of control facilities and audit issues in the predefined environment.

The analysis revealed that each layer of the UNIX operating system as defined in the model can act as a screen preventing the users from getting to the next layer.

Using the methodology established in this research essay, and the Access Path and Path Context Models, it was possible to evaluate the predefined UNIX environment and develop an audit programme for the review of a UNIX installation.

It was found that in each of the different layers, except for the networking layer of the predefined path there was sufficient controls to ensure that:
- access overall are restricted to system objects on a per user or per group basis, as well as controls to limit the propagation of access rights;
- users are only allowed on networks after they have uniquely identified themselves; and
- An audit trail of all access of system objects are maintained.

Because of the impact of the security weaknesses of the network layer on the rest of controls in the system, it was found that the operating system have more risk factors than controls to control these risks. It is therefore concluded that all control objectives will not be met by the UNIX operating system if any of the components in table 5.1 are active on the system without the use of an external computer security package to compliment the controls of the UNIX operating system.

Because each layer in the model consists of different components, all of which do not need to be active for the UNIX operating system to function properly, the auditor must consider whether the components active on the installation under review are required and the effect thereof on the overall computer security, as some of these components present the installation with risk factors that cannot be effectively and efficiently countered by any of the layers in the model.

1.9 Conclusion

The starting point of computer security within an organisation is the security policy laid down and enforced by management. The security policy contains management security expectations based on the risks and threats that the organisation faces.

In modern computer installations implementing and maintaining controls can be a daunting task as the security features of modern installations can be very complex. The Access Path and Path Context Models (Boshoff 1985, 1990) supply a method whereby the security features in a complex computer installation can be reviewed, evaluated and concluded on.
## CHAPTER 2

ESTABLISHING THE FRAMEWORK - THE MODEL FOR THE UNIX ENVIRONMENT

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 INTRODUCTION</td>
<td>10</td>
</tr>
<tr>
<td>2.2 OVERALL ORGANISATIONAL POLICY</td>
<td>11</td>
</tr>
<tr>
<td>2.3 UNIX CONCEPTS</td>
<td>13</td>
</tr>
<tr>
<td>2.4 COMPONENTS IN THE UNIX OPERATING SYSTEM</td>
<td>14</td>
</tr>
<tr>
<td>2.5 ESTABLISHING THE FRAMEWORK</td>
<td>20</td>
</tr>
<tr>
<td>2.6 CONCLUSION</td>
<td>26</td>
</tr>
</tbody>
</table>
2.1 INTRODUCTION

In recent years the rapid development and improvement of computer facilities have led to unprecedented growth in the use of computer systems to communicate, process and store information. Data processing and information systems have become critical components in ensuring the continued operation of an organisation's business activities.

In today's world of networked workstations and minicomputers every UNIX system is a potential target for breaches in computer security. As business is relying on computer systems to deliver error free information the need for controls over complex computer systems has increased.

UNIX was originally developed for a research and development environment where security was not a priority. In fact, access and data security measures were often a hindrance in this field as data and resources were often shared between users in this field.

The long and sad history of UNIX security weaknesses has been well documented, creating a widespread impression that UNIX cannot be secured. However, as UNIX became more prevalent within commercial applications, a new emphasis on security emerged and major modifications were made to UNIX systems to add security provisions which were not designed into the original product.

This chapter will describe some fundamental UNIX concepts that are necessary to gain an understanding of how the UNIX operating system functions. A framework under which the Path Context Model will be applied to the UNIX operating system will then be developed. The environment will be defined in terms of the model and a description of the environment and its components will be developed. The definition and description will concentrate on aspects deemed more important and relevant to security, and that will be used to evaluate the predefined control objectives.
The objective is to develop a system of controls and procedures that are aimed at restricting access to computer resources.

2.2 OVERALL ORGANISATIONAL SECURITY POLICY

If computer systems and networks are used by different people within an organisation such an organisation needs a security policy. A security policy educates users, garners support from management, and sets standards for the proper behaviour of users and system administrators.

The framework for the UNIX system must not be seen in isolation, but rather as a part of the overall organisational security policy.

According to the BSI (1993:10) creating and defining a framework for overall organisational security consists of three basic requirements:

- The first source of security requirements is the unique set of security risks to assets and their potential impacts on business;
- The second source of requirements is the set of statutory and contractual requirements networking proliferates; and
- The third source of requirements is the unique set of principles, objectives and requirements for information processing that the organisation has developed to support their own business operations.

It is important that the organisation security policy supports these requirements, and vital that the implementation of security controls in the IT infrastructure does not present any major obstacles in achieving efficient business operations.
The framework being developed for the UNIX operating system must be evaluated in connection with the above mentioned and the following will be considered:

- The capabilities of system software to implement the security philosophy;
- The evaluation of controls available in the various system software components that can achieve the various control objectives. The concept of different layers of system software being used to control and prevent users from gaining unauthorised access to computer systems will be expanded upon in Chapter 3 and Chapter 4 an audit programme will be developed to evaluate if the identified components in each layer will achieve the control objectives set out in Chapter 1; and
- Concluding on the capability of system software to provide controls to ensure use of system software controls.

2.2.1 Critical Success Factors for an Overall Security Policy.

According to the BSI (1993:12), it has been shown that the following factors are often critical to the successful implementation of information security within a company:

- Security objectives and activities must be based on business objectives and requirements, and led by business management;
- There must be visible support and commitment from top management;
- There must be a good understanding of security risks to the organisation, and the level of security inside the organisation;
- Security must be effectively marketed to all managers and computer users within the organisation; and
- Comprehensive guidance on security policies and standards must be distributed to all employees and contractors.
2.3 UNIX CONCEPTS

2.3.1 Introduction

UNIX, like other operating systems, is a layer between the hardware and the applications that run on the computer. UNIX is therefore responsible for managing and running the system providing services to users and applications.

UNIX includes all the traditional operating systems components as well as a set of libraries and a set of applications. The components and layers of UNIX can be set out as depicted in figure 2.1.

UNIX provides services to users and applications by managing input/output devices, memory, processes and the file system through the use of a kernel. All tasks in UNIX are performed by processes. These processes are managed by the kernel and make use of hardware independent interfaces to address memory and input/output devices (Armstrong, et al 1994:7).
The UNIX system views all hardware devices as a file. This allows the same simple method of reading and writing files to be used to access these hardware devices.

All files on a UNIX system are stored in the file system that plays an essential role in the UNIX operating system. The file system manages read and write access to user data and to devices, such as printers, attached to the system. The file system implements security controls to protect the safety and privacy of information (Armstrong, et al. 1994:7).

As users have only access to the standard libraries and applications, many people think that these two components are UNIX. These two components however constitute the UNIX interface.

The standard applications like grep allows users to search through an entire file for a specified regular expression and display the line or lines that contain that expression. WC or word count allows the user to do a word count of a specific file. These are only two of the hundreds of utilities that are supplied with UNIX.

2.4 Components in the UNIX operating system

The major components in the UNIX operating system are:

- The Kernel;
- Shells;
- UNIX Processes;
- UNIX File Structures; and
- UNIX utilities and user programs.

The structure of UNIX can be set out as depicted in figure 2.2. The major components of the UNIX operating system as identified in section 2.4 will be discussed in paragraph 2.4.1 to 2.4.5.
2.4.1 The Kernel

The kernel forms the heart of the UNIX operating system. It manages all processes running on the system and the system hardware while also performing various low-level and system-level functions. The kernel interacts directly with the system hardware and isolates other parts of the UNIX system from hardware dependencies. The kernel is also responsible for running and scheduling processes by prioritising and scheduling multi-user operation of the central processor and managing main memory by allocating it among user processes. The kernel is also responsible for maintaining accounting logs of system activity and usage (Ernest & Young, 1994:17).

At the lowest level, the kernel communicates directly with the hardware. Parts of the kernel, therefore, must be custom-made for each type of hardware. Other parts of the kernel are hardware independent (Armstrong, et al 1994:7).
The kernel is loaded into memory when the computer is first turned on. Once the kernel is loaded into memory it is ready to carry out requests made by users and other UNIX utilities. Requests by users or utilities to the kernel are done by way of system calls. Each system call instructs the kernel to perform one particular service on behalf of the program making the call. The UNIX kernel supports many different system calls, each being independent of the underlying hardware. Therefore application programs, using only standard system calls, can be ported to other UNIX implementations very easily.

2.4.2 Shells

Instructions made to the kernel are of a critical nature complex and highly technical. To protect users from the complexity of the kernel, and to protect the kernel from the shortcomings of the users, a protective shell is built around the kernel. The shell is therefore the command interpreter for the UNIX system. The shell accepts user commands, interprets them and passes them onto the kernel. The shell is a sophisticated utility program that functions as the primary user interface to the UNIX system (Anderson, Costales & Henderson, 1991:23).

The shell supports both interactive and background processing. Interactive processing uses direct communication between the user and the UNIX system. Background processing allows non-interactive tasks to continue while the user continues with other interactive processing (Ernest & Young, 1994:17).

To function as a user interface the shell has to perform the following basic functions (Armstrong, et al 1994:7):

- Command line interpretation - Input to a shell prompt is sometimes called a command line. The basic format for a command line is: command arguments;
- Program initiation - When the shell finishes interpreting the command line, it initiates the requested program by searching for the executable file in the directories specified by the PATH environment variable. When the executable file is found, a subshell is started for
the program to run. A subshell can establish and manipulate its environment without affecting the environment of its parent shell;

- **input/output redirection** - The shell allows input to be redirected from other sources, such as a file or tape, and output to be redirected to other destinations such as a file or a printer;
- **Pipeline connection** - Pipeline connections are a special kind input-output redirection in which the standard output of the one command is piped directly into the standard input of the next command;
- **Substitution of file names**;
- **Maintenance of variables** - The shell is responsible for maintaining variables for later use;
- **Environment control** - When the login programs invokes the user's shell, it sets up the user's environment, that includes the user's home directory, the type of terminal used and the path that will be searched for executable files; and
- **Shell programming** - The shell is also a programming language. By combining commands and variable assignments with flow control and decision making, the shell forms a powerful programming tool.

### 2.4.3 UNIX Processes

When a program is executed in the UNIX system, the system creates a special environment for that program. This environment contains everything necessary for the system to run the program as if no other program were running on the system (Armstrong, *et al* 1994:728).

Every time a program is executed the UNIX system does a *fork*, that performs a series of operations to create a process context and then executes the programs in that context. The steps include the following (Armstrong, *et al* 1994:728):

- **Allocate a slot in the process table**, a list of currently running programs kept by UNIX. UNIX creates the impression of multiple programs running simultaneously by quickly switching between processes in the process table;
- **Assign a unique process identifier (PID) to the process**. This identifier can be used to examine and control the process;
• Copy the context of the parent, the process that requested the spawning of the new process; and
• Return the new PID to the parent process. This enables the parent process to examine and control the process directly.

The fork copies the current processes in two identical processes. The original process is called the parent, and the newly created process is called the child. After the fork the child process will have the same characteristics as the parent, and as a result has the same owner and group associated with it (Armstrong, et al 1994:729).

The fork performs as series of operations to create a process. This series of operations include the following (Armstrong, et al 1994:7):

• Every process gets allocated a slot in the process table;
• Every process is assigned a unique process identifier (PID) by which the process can be controlled and examined;
• Copy the context of the parent process; and
• Return the new PID to the parent process.

After the fork is complete an exec is called. An exec overwrites the executable code of the child process with the executable code of the new process being created. Subsequently the parent process (i.e. the shell) is put to sleep and the execution of the child (i.e. application program) process can be begin (Garfinkel & Spafford, 1993:428).

Because of this procedure all processes initially have the same characteristics and rights as their parents.

Daemons are programs that execute and operate in the background. It is a process that performs a certain task for the operating system on a regular basis. For example the printer queue daemon handles the processing of printer output (Ernest & Young, 1994:26).
2.4.4 UNIX File Structures

The UNIX operating system is organised as a hierarchical file structure, that is, as an ordered relationship of files and directories. Figure 2.3 shows directories in a typical UNIX system.

![UNIX Directory Structure Diagram](image)

**Figure 2.3: Typical UNIX directory structure** (Anderson, Costales & Henderson, 1991:36).

UNIX directories can contain files, symbolic links, and other directories. Each directory entry contains the file name and a pointer to the information node (inode) of a file. The inode in turn contains pointers to the disk blocks being used by the file on the hard disk. An inode typically contains the following information pertaining to a file (Ernest & Young, 1994:39):

- The file's owner;
- File permissions;
- The file directory;
- The file's type;
- The file's size in bytes;
- The time the file's inode was last modified;
- The time the file's contents was last modified;
The time the file was last accessed; and
- Reference count: the number of names the file has.

Most UNIX systems have standardised the main directories in which the system and user files are stored (Ernest & Young, 1994:39).

2.4.5 UNIX Utilities

Numerous utility programs are supplied with UNIX. These utilities support a variety of tasks, such as copying files and developing software. Utilities can only modify files according to the file permissions of the file being modified. Utilities should therefore not impose additional security risks. If the security on a system is poorly defined, utilities can offer users the opportunity to exploit the poor security for their own benefit.

2.5 ESTABLISHING THE FRAMEWORK IN THE PREDEFINED ENVIRONMENT

2.5.1 Introduction

Computer systems are perceived by users to be transparent due to the fact that users normally log into the system and then automatically receive the data or application program requested by them at the terminal they are working at.

Figure 2.4: User View
In reality the user has to pass through many elements of the system software before gaining access to application programs and or data. If a user wants to use an application program or certain data, his request is passed through different layers of systems software.

In the remainder of this chapter the different components of the model and their functions will be described. Chapter 3 will cover each of the components in more detail.

2.5.2 Boot Process

The bootstrap program is the first program to run on the server when it is “booted”. Although every UNIX system’s bootstraps in a slightly different fashion, usually the Read Only Memory Monitor loads a small program called “boot” that is kept at a known location on the hard disk or network. The boot program calls and load the UNIX kernel into the computer and starts its running (Ernest & Young, 1994:22).

2.5.3 Kernel

After the kernel initialises itself and determines the machine’s configuration, it creates a process with a PID of 1 and runs the `etc/init` program (also known as the init daemon). The kernel is the main system process that controls the system resources and all other processing on the server. Unlike the bootstrap the kernel is never replaced (KPMG Peat Marwick 1994:5).

2.5.4 Init

Once the kernel is loaded into memory it is ready to carry out user requests. First though, a user must log in to make a request. For a user to log in, the kernel must know who the user is and how to communicate with him. To do this the kernel first invokes the init daemon. The init daemon is the parent of all other processes and daemons on the system. UNIX will not function without the init. One of the most important process spawned from the init is the getty daemon. The init process sets many processes in motion, based on the initialisation state defined in the `/etc/inittab` file (Armstrong, et al 1994:1050).
2.5.5 **Getty**

For every user port (usually referred to as tty) the kernel invokes the getty program that starts communication with all the terminals on the network. This process is called spawning. The getty program displays a login prompt and continuously monitors the communication port for any type of input that it assumes is a user name. Figure 2.5 shows a freshly booted UNIX system with 8 user ports (Armstrong, et al. 1994:347).

![Figure 2.5: An active system with no users. (Armstrong, et al. 1994:347).](image1)

A user sitting at a terminal will enter a login ID at the login prompt received from the getty. The getty program calls the login program after input from the user is received. This is shown in figure 2.6.

![Figure 2.6: A User Logs In (Armstrong, et al. 1994:347).](image2)
2.5.6 LOGIN

The login program establishes the identity of the user and validates his right to login. The login program checks the password file. If a user fails to enter a valid password, the port is returned to the getty. If the user enters a valid password, the login program performs some account initialisation tasks then changes the effective user id (UIDs) to be those of the user name that has been supplied (Armstrong, et al. 1994:346). Login passes control by invoking the program name found in the user's entry in the password file. This program might be a word processor or a spreadsheet, but usually is a more generic program called a shell (Figure 2.7).

2.5.7 Shell/Applications

The user's command shell or start-up application is invoked by the login program and control is passed to the user. Figure 2.7 depicts users that have control passed to them either through a shell or an application program.

![Diagram of an Active UNIX System](image)

**Figure 2.7: An Active UNIX System** (Armstrong, et al. 1994:348)

Files that log a user's movements on the system are updated to reflect that the user is on the system. UNIX next reads the file that sets the environmental settings for all users. It then identifies the file that sets user specific environmental setting for the user that logged into the system. After all these processes UNIX hands control to the user. This can be control of started applications or control of a shell.
The above mentioned description is graphically depicted in figure 2.8

Figure 2.8 The Boot Process (Ernest & Young, 1994:22)
The above mentioned steps can further be divided into the following layers that a user of the computer has to "travel" through to get the application or data that has been requested by the user. The user "travelling" through the different layers of the UNIX operating system is depicted in figure 2.9.

![Figure 2.9: UNIX Layers](Ernest & Young, 1994:22)

The grouping off the components mentioned and depicted in section 4 of this chapter into the above layers and the security feature's for each layer will be discussed in detail in Chapter 3.
2.6 CONCLUSION

Each level in the access path provides some form of integrity or security checking, or at least the opportunity for controlling who has access to what system resources.

The controls that will be needed to provide adequate security to meet the control objectives are likely to be a combination of the various levels of the access path. The system software works together to provide a controlled environment.

The access path depicted in figure 2.9 shows the different layers of the UNIX operating system that will be dealt with in Chapter 3. The major and more common components in the different layers will be discussed in Chapter 3.
3.1 INTRODUCTION

Identifying potential risks and controls in a complex and sophisticated operating system such as UNIX can be a daunting task.

The simplest, easiest and most effective way of identifying potential risks and evaluating the controls to limit the dangers posed by these potential risks, is to use the Access Path Model.

In using this approach the security measures of each component or layer that are invoked by the operating system are documented, reviewed and audited in the same way a user attempts to access the data.

The different components of the UNIX logical access path was described and graphically depicted in figure 2.9. Each component of the assumed environment will be dealt with below. The detail workings of each component will not be dealt with. Focus will be placed on key parameters and files that will affect security.

3.2 NETWORK AND COMMUNICATION CONTROLS

3.2.1 Introduction

UNIX is a networking operating system. It is tightly integrated with the TCP/IP networking protocols. UNIX has a well developed and rich set of networking utilities available to both the user and the system administrator.

Network and communication controls are first “layer” of controls perceived by users of the system. This is a very complex area and encompasses many different situations. It does not fall in the scope of this essay to discuss every possible situation and therefore emphasis is placed on the more common utilities and features of UNIX that is responsible for networking and communication.

UNIX offers a number of network services including the following (Garfinkel & Spafford, 1993:223):

- Remote virtual terminals - rlogin and telnet that allows users to remotely connect to other computers on the network;
Remote file service - Allows users to remotely access their files on another UNIX computer on the network;

Electronic Mail - Allows users to send messages to other users on the network; and

Electronic directory service - finger and whois that allows one to obtain key information on other users of the network such as the user name, telephone number and so on.

3.2.2 Terminal Access

Most network services under UNIX are not handled by the UNIX operating system itself, but by programs that run automatically. These are called daemons. Three of these daemons can be seen as the controllers. There is one daemon for each of the communication service types in UNIX namely (Armstrong, et al, 1994:254):

- inetd - for sockets;
- rpcbind - for RPC; and
- listen - for SAF.

3.2.3 The Master Daemon - inetd

Rather than having each task listen for connections on its ports, UNIX uses a common daemon to listen on many ports at once. This is the Internet services daemon or inetd. It listens to every port listed in its configuration file (/etc/inetd.conf). When it receives a connection it forks off and starts the appropriate service daemon. Inetd handles both TCP and UDP servers. UDP is the simpler of the two and is an unnumbered message sent to a particular IP address and port. UNIX buffers the request and provides the message to any process that reads that port. It is a connectionless service and no acknowledgement of reception is sent to the sending system (Armstrong, et al, 1994:1159). Only complete messages can be read.

TCP is a connection-orientated protocol that guarantees delivery of data free from errors and in the correct order. A TCP connection is a unique fusion of four values (Garfinkel & Spafford, 1993:227):

- The sending IP address;
- The sending port number;
- The receiving IP address; and
- The receiving port number.
This allows multiple connections at the same time to the same receiving port, as all four values uniquely identify a connection. The connection is bi-directional, and what is written at the one end is read by the other, and vice versa (Garfinkel & Spafford, 1993:223).

**Inetd** is run at boot time by `/etc/rc`. When inetd is started, it probes the contents of the `/etc/inetd.conf` file to ascertain which network services it is supposed to handle. Inetd uses `bind(2)` to attach itself to different network ports and then uses the `select(2)` system call to cause notification when a connection is made on any of the ports. The `/etc/inetd.conf` file contains the following information (Garfinkel & Spafford, 1993:232).

- **Service** - The name of the service as listed in the `/etc/services` file;
- **Socket type** - Whether the service expects to communicate via a stream or a datagram. Options are `stream` for TCP services or `dgram` for UDP;
- **Protocol** - TCP and UDP are the only protocols supported;
- **Wait** - Indicates whether **inetd** should wait until the server exists before listening again, or listen right away. Options are, `wait` - the server is expected to process all subsequent datagrams received on the socket or `nowait` - where **inetd** will fork(2) and `exec(2)` a new server process for each additional datagram or connection request received;
- **User ID** - Specifies what user ID **inetd** should use when starting the server. This can be root (UID 0), daemon (UID 1), nobody (-2) or an actual user of the system. This field allows server processes to be run with fewer permissions than root, to minimise the damage that could be done should a security hole be discovered in a server program;
- **Process to run** - The file name to run; and
- **Command String** - The remainder of the line is passed in as the command string.

### 3.2.4 The `/etc/services` File

The `/etc/services` file contains a list of all the network service that UNIX can implement. Each line of the file contains the following information (Garfinkel & Spafford, 1993:230):

- **Service Name**;
- **Network Port Number**;
- **Protocol Name**; and
- A list of Aliases

### 3.2.5 Telnet

Remote serial communication via IP is performed through telnet. The receiving UNIX system listens on TCP port 23 for any telnet connections that are made via the `telnetd` or `inetd`
daemons. Once a connection is established, the \texttt{telnetd} daemon establishes a connection to the \texttt{login} program and a remote login session is started (Armstrong, et al., 1994:1135).

Using telnet can pose great security risks to computer systems. This risk is caused by the manner in which some networks are implemented. On many networks, the packets sent between computers are delivered to every computer that is linked to the physical piece of wire. Computers are normally programmed only to listen for packets that are intended for them. Computers however can be reprogrammed to listen to every packet that is transmitted through the physical piece of wire they are connected to. Special programs can capture the first few characters sent in both directions on a telnet connection and thereby capture the user's password and ID (Garfinkel & Spafford, 1993:234).

3.2.6 Rlogin, Rlogind and Rsh

The \texttt{rlogin} and \texttt{rlogind} applications establish a remote terminal session between two computers connected on the network. When \texttt{rlogin} is successfully executed, the shell prompts and commands entered on the local computer, are executed on the remote computer (Armstrong, et al., 1994:263).

The \texttt{rlogin} command takes a mandatory argument that specifies the remote host. Both the remote hosts and the local host must have \texttt{rlogin} available for a connection to be established.

During a \texttt{rlogin}, the \texttt{rlogin} protocol takes control of the identification and authorisation procedure normally performed by the \texttt{login} program. This \texttt{rlogin} protocol initiates the login session on the remote host for a particular user. By default the user is the same as the local user and therefore it is not necessary for the user to type in his user name (Garfinkel & Spafford, 1993:235).

The \texttt{rlogin} protocol also lets users log in to another computer if the connection is made from a trusted host or user.

\texttt{Rsh} is similar to \texttt{rlogin} except that instead of logging the user in, it allows the user to run a single command on the remote system. \texttt{Rsh} only works from trusted hosts or trusted users.

3.2.7 Trusted Hosts and Users

A trusted host is another UNIX system in which the user has "trust", and thus allows its users to login to a UNIX system without entering a password. In this system users first have to login into their "home" system, and are therefore still subjected to the password and access security restrictions that are enforced at their "home" system, before access can be
gained to the trusted host. Each UNIX system therefore “trusts” the security of the other ensuring that only valid users gain access to the network (Garfinkel & Spafford, 1993:235).

Trusted users are like trusted hosts, except they refer to individual users, not hosts. If a user is designated as a trusted user on another computer for a specific user’s account then that trusted user can log into the user’s account without typing a password.

When either the rlogin or rsh commands are issued to a networked UNIX host, that host’s rlogin or rshd programs scan the /etc/hosts.equiv file to validate that a trusted connection can be made. After the /etc/hosts.equiv has been scanned the rlogin and rsh programs scan the .rhosts file in the user’s home directory. The .rhosts file allows users to build a set of trusted hosts applicable only to that user. A user’s .rhost file can also contain host and user name pairs extending trust to other user names. If the /etc/hosts.equiv and the user’s .rhost files do not have the necessary entries for a user’s host and user name, a password must be entered before access can be gained (Ernest & Young International Ltd 1994:49).

Users’ .rhost files are easily utilised for unintended purposes. Users are allowed to freely work between two or more computers using the rsh command. Users can also make their account available to other users without revealing their password. By the same token an unauthorised user who has access to another user’s .rhost file may be able to add their own user name making it easier to gain access in the future (Garfinkel & Spafford, 1993:237).

3.2.7.1 The /etc/hosts.equiv File

The /etc/hosts.equiv file contains a list of all the trusted hosts for a user’s computer. Any host name listed in this file is treated as a trusted host, therefore a user who connects with rlogin or rsh from that host will be allowed to log in or execute a command from a local account with the same user name without typing a password (Garfinkel & Spafford, 1993:236).

3.2.8 UNIX TO UNIX COPY PROGRAM (UUCP)

UUCP is a package of commands that allows a user to transfer data and files between UNIX systems and execute commands on a remote system across a serial port, usually via a modem.


- uucp, that copies files between computers and
- uux, that executes programs on remote computers.
3.2.8.1 How UUCP works

UNIX systems that are running uucp will, at a specific time or when required to, place a telephone call to another UNIX system. The time and the system to be connected to will be determined by a scheduling daemon. The uucico daemon will be invoked at the specified time to start and carry out communications (Ernest & Young, 1994:35).

When invoked uucico scans the permission file to determine which commands the remote machine can execute and which files can be accessed. The permission file contains the following information (Ernest & Young, 1994:35):

- lists of other systems than can be contacted;
- other systems telephone numbers;
- the local devise to use;
- the relevant baud rates to use;
- the login ID to be used; and
- the password for that site.

The login will access the pertinent system and the remote system will invoke its copy of uucico. The host uucico will intercept the login and confirm that it can communicate with the system. After verification has been completed, the two uucico daemons will communicate and perform the tasks required. When communication is halted the uucico daemons will log all details of the communication in their log files (Garfinkel & Spafford, 1993:196-197).

3.2.8.2 Security and UUCP

Any computer system that allows files to be copied from computer to computer and allows commands to be remotely executed raises a number of important security issues and concerns. UUCP has however many security measures built into it to reduce the threats and dangers posed by its capabilities.

Because uucico logs into remote systems, it has to keep track of all the names, telephone numbers, account names and passwords it uses to log into these computers. This information is kept in a special file called /usr/lib/uucp/L.sys or /usr/lib/uucp/Systems (Ernest & Young, 1994:52).

The information in these files can easily be misused and file permissions should be set to read and write only for the owner.
3.2.8.3 The /usr/lib/uucp/L.cmds file

Commands that can be executed from a remote system via uucp should be limited to ensure that other users cannot delete, modify or copy files on computer systems. For this reason, uucp allows one to specify which commands remote systems are allowed to execute on one's computer. The /usr/lib/uucp/L.cmds file contains the list of valid commands that can be executed by a remote system, on the host system. Commands not listed in this file cannot be executed by uux. The /usr/lib/uucp/L.cmds contains only two commands that are mandatory namely rmail (the receive mail command) and the rnews (the receive news command) (Ernest & Young, 1994:52).

3.2.8.4 The /usr/lib/uucp/SCIFILE file

This file contains information that is used to perform sequence checks with other systems. These sequence checks are before hand agreed upon by systems administrators and uucp is set-up accordingly. Every line in this file contains information pertaining to the last communication session with that system (Ernest & Young, 1994:52-53).

The system does a sequence check by checking if the information on your system corresponds to the information on the other system. If this information does not agree the communication will fail. This checking is performed to stop unwanted access by a system masquerading as an authorised system. As with the L.sys file, this file should be owned by the uucp user and groups and file permissions should be set to those of the owner which in this case should be read and write only (Ernest & Young, 1994:53).

3.2.8.5 The /usr/lib/uucp/USERFILE file

The /usr/lib/uucp/USERFILE file controls the file and directory access permissions pertaining to the use of uucp programs on the local system by local users and other systems. Normally, there is one entry for each uucp login in the /etc/passwd file. Entries can also be included for individual users on the system to give additional uucp privileges. A USERFILE entry can specify four things (Garfinkel & Spafford, 1993:201):

- Which directories can be accessed by remote systems;
- The login name that a remote system must use to talk to the local system;
- Whether a remote system must be called back by the local system to confirm its identity before communication can take place; and
- Which files can be sent out over uucp by local users.

The configuration of the /usr/lib/uucp/USERFILE file is very complex and mistakes in this file can cause uucp not to work at all or can cause security holes.
The structure of the entries in this file is `user name, system-name [c] path names(s)`. This structure is explained in table 3.1.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>Function in the USERFILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>user name</td>
<td>Login name in <code>/etc/passwd</code> that will be used.</td>
</tr>
<tr>
<td>system name</td>
<td>System name of the remote system.</td>
</tr>
<tr>
<td>c</td>
<td>Optional call-back flag. If this flag is present uucico on the local computer halts conversation after the remote machine called the local machine. Uucico on the local machine then calls back the remote machine in order to establish its identity.</td>
</tr>
<tr>
<td>path name</td>
<td>List of absolute path name prefixes separated by blanks. The remote system can only access those files beginning with these path names. A blank field indicates open access to any file in the system.</td>
</tr>
</tbody>
</table>

Table 3.1: Structure of the `/usr/lib/uucp/USERFILE`

The security measures to be followed with this file are as follows:

- The file must be owned by the uucp user and group. File permissions must be set to read and write only for the owner;
- Great care must be taken when specifying to which directories the remote system can have access. For instance the remote user should not have access to the `/etc` directory, as this contains the `passwd` file and to the `/usr/lib/uucp` directory because users can then change the USERFILE or the `Lcmds` files to allow them to execute any command they wish;
- File specific to remote systems should be kept in separate directories where only those systems have access;
- Giving all access from the `/ (root)` directory is also dangerous because this makes it possible for people outside your organisation to subvert your system easily, as they can then modify directory on your system that is world-write-able; and
- The home directory for the uucp user should not be in the directory `/usr/spool/uucp/uucppublic`, or any other directory that can be written to by a uucp user.

### 3.2.8.6 UUCP Daemons

There are four uucp daemons that should be invoked on a regular basis. They are the `admin`, `cleanup`, `polling` and `hourly` daemon. All these daemons are started from the cron.
3.2.8.7 The admin Daemon

The admin daemon will give, by e-mail, a brief image of the state of uucp, including a snapshot of the running processes and a listing of the job queue. It will also scan the log files to see if there were any attempts to transfer the passwd file (Armstrong, et al, 1994:1404).

3.2.8.8 The cleanup Daemon

The cleanup daemon will backup up all the log files and save them for three days. The cleanup daemon will invoke the uucleanup command. This command removes old jobs from the queue based on line arguments (Armstrong, et al, 1994:1405).

3.2.8.9 The polling Daemon

This daemon examines the poll file to create polling requests for uucico. This is essentially touching a file in the spool directory (Armstrong, et al, 1994:1405).

3.2.8.10 The hourly Daemon

The hourly daemon runs the uusched command, that examines the spool to find any queued jobs, and if it finds jobs, it runs uucico for that system. When it finishes, it runs uuxqt to execute any incoming jobs (Armstrong, et al, 1994:1405).

3.2.9 FTP

The File Transfer Protocol allows users to access a UNIX system and to transfer complete files between systems. When a user uses FTP to contact a remote machine, the remote machine requires that the user log in by providing one’s user name and password. By default FTP uses the /etc/passwd file for identification and authentication. All FTP logins are recorded on the remote computer in the /usr/adm/wtemp file (Ernest & Young, 1994:60).

Although FTP only allows users to transfer files, these files’ accessibility are still governed by the permission bits on these files.

Because FTP requires passwords to be typed and then to be transmitted over the network, these passwords can be intercepted as in the case of telnet and rexec. Further in many cases, end-users after logging on are directed to their own start-up applications. Normally they would only have access to read and modify files that can be accessed by their applications. By granting these users access to FTP, it is possible that they can bypass the
normal route that directs them to their applications, thereby accessing files directly (Ernest & Young, 1994:60).

Because of the above mentioned it is advisable to restrict access to FTP to those users that have explicit need for it. Users with extensive rights should not be allowed to use FTP. Users not allowed access must be defined in the /etc/ftpusers file.

### 3.2.9.1 Anonymous FTP

FTP can be set up for anonymous access that will allow people on the network who does not have an account on one’s machine to deposit or retrieve files from a special directory. This capability of FTP is normally used by software suppliers to gain quick access to software programs to make small modifications, etc.

An anonymous FTP facility should only be provided with the following security measures in place (Ernest & Young, 1994:61):

- Create a special account with the name ftp and a home directory outside the normal user directory structure. Users should not have write access to this directory;
- Create bin and etc sub directories of the ftp home directory and copy only those utilities that are necessary. Also transfer to the new etc directory copies of the passwd and group files. These files should be modified so that it only contains ftp user details. Both the new bin and etc directories should be owned by root and users should not have write access to these directories;
- Create a sub directory owned by root and not writable by any user. Files that are publicly available should be placed here; and
- Remote users can transfer large files to one’s system, thus denying access to the system. A file quota should be placed on a user using ftp, or a home directory in an isolated partition should be allocated.

### 3.2.10 Finger

The finger program has two uses (Garfinkel & Spafford, 1993:240):

- If the program is run with no arguments, the program prints the user name, full name, location, login time, and office number of every user currently logged onto the system if the above mentioned information is stored in the /etc/passwd file; and
- If the program is run with a name argument it scans the /etc/passwd file and prints detailed information for every user with a first, last, or user name that matches the specified name.
The `/etc/fingerd` program implements the finger protocol, that makes `finger` available to anyone on the network.

`Finger` poses a potential security risk as it makes it easy for intruders to determine who is a valid user on the system.

3.2.11 The `rexec` Execution

The `/etc/rexec` program allows users to execute commands on other computers without logging into these computers. This program does not use the trusted host concept and so a user can use it to execute a command on any network host computer (Garfinkel & Spafford, 1993:239).

The client opens up a connection and transmits a message specifying the user name, the password and the name of the command to execute. As `rexecd` does not use the trusted host mechanism, it can be used from any host on the network. However, because `rexecd` requires a password to be sent over the network it is susceptible to the same password snooping as `telnet`.

3.2.11 Network Information Service (NIS)

NIS is a distributed database system that allows computers to share passwords, files, group files, hosts tables and other files other a network. These files are kept on a single computer normally called the NIS master service. From a users' perspective it will seem that these files reside on all computers on the network (Ernest & Young, 1994:56).

NIS works by having a special line in a system database file (for example `/etc/passwd` or `/etc/group`) that begins with a "+" sign. The "+" sign tells the UNIX programs that scan that database file to ask the NIS server for the remainder of the file (Garfinkel & Spafford, 1993:256).

NIS makes it easier to manage large networks because all the account and configuration information are stored on a single machine. Care should be taken so that the "+" sign in the `/etc/passwd` file is in that of the clients and not the server (Garfinkel & Spafford, 1993:257).

3.2.11.1 Netgroups

Netgroups forms a part of NIS, and is a system for classifying users and machines on a NIS network (Garfinkel & Spafford, 1993:257).
Netgroups can be used to restrict or specify who is allowed to make use of certain services. By properly specifying netgroups, the security of systems can be increased by limiting the machines and users that have access to critical resources. The netgroup database is kept on the NIS master in the `/etc/netgroup` or `/usr/etc/netgroup` file.

3.2.12 Network File Systems (NFS)

Network file system allows individual computer systems to share files over the network. Unlike other remote file systems, NFS allows users to change and read the contents of files stored on the server without ever having to log in or supply a password (Garfinkel & Spafford, 1993:259).

NFS is based on two protocols namely MOUNT and NFS. The NFS server uses the MOUNT protocol to identify which file systems are available and to which hosts it is available. The NFS protocol is used to make the files available to clients (Garfinkel & Spafford, 1993:263).

The `/etc/dfs/dfstab` file designates which clients can mount the server’s file system and what access is given to those clients. The following access options can be set in this file (Ernest & Young, 1994:60):
- `ro` Directory and contents are exported/shared in read only format;
- `rw = ....` The hosts listed after `rw` have read and write capabilities. If no hosts are listed then all hosts have read and write capabilities; and
- `access =` Access is only granted to those hosts listed after access.

With the showmount command NFS usage and activity can be monitored. This command lists all the clients that have mounted directories from the server. This command however does not tell one which hosts are actually using your exported file systems, it only lists those hosts that have mounted the file system (Garfinkel & Spafford, 1993:264). For continual monitoring, a shell script using `cron` and `awk` could check that all hosts are authorised, and if not so, inform root or the system administrator of the presence of an unauthorised host.

3.3 ACCOUNTS / USER SECURITY

3.3.1 Introduction

The second layer in the access path focuses on user definition, authentication and password controls. The granting of authority for a user to access the system and its resources forms a fundamental component of user security. This action is performed manually by management and does not form part of the UNIX operating system. It also falls outside the scope of this essay.
3.3.2 User Accounts

Every user who uses a UNIX system should have a unique account. A UNIX account is made of two main parts:

- A user name and
- A password.

3.3.2.1 The /etc/passwd File

From a security point of view the /etc/passwd file is the UNIX operating system's most important file. If a user can modify the contents of this file, a user can become the "superuser" by changing his/her identifier (UID) to "0".

UNIX uses the /etc/passwd file to keep track of every user on the system. The /etc/passwd file contains the following information (Garfinkel & Spafford, 1993:23):

- User name;
- Encrypted password;
- User identification number;
- Group identification number;
- A descriptive field - known as the GCOS field;
- Home directory; and
- Shell script.

Each line in the file contains a database record - the record field is separated by a colon (:).

An example /etc/passwd file is depicted below in figure 3.1.

```
root:ML:xx43c0m17Ee:0:1:System Administrator:/:  
daemon:*:1:1:sys:/usr/src:  
sys:*:2:2:bin:/usr/src:  
bin:*:3:3:adm:/usr/adm:  
adm:*:4:4:uucp:/usr/lib/uucp:  
uucp:*:5:5:uucp administrative login:/usr/lib/uucp:  
nuucp:*:6:6:uucp network login:/usr/spool/uucppublic:/usr/lib/uucp/uucico:  
lp:*:7:7:lib:/usr/lib:  
jenneri:ZLHIchPvW3uKk:100:100:A/c clerk:/appin:/bin/sh:  
suzan:UrXhXzHIX2x20:100:100:Payroll mgr:/appin:/bin/sh
```

Figure 3.1: Example /etc/passwd File
The above example can be explained as follows in table 3.2:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>Login Account name</td>
</tr>
<tr>
<td>xx43c0m17Ec</td>
<td>Encrypted password</td>
</tr>
<tr>
<td>0</td>
<td>User ID reference number for this login</td>
</tr>
<tr>
<td>1</td>
<td>Group ID reference number for this login</td>
</tr>
<tr>
<td>System administrator</td>
<td>Descriptive text field (optional)</td>
</tr>
<tr>
<td>lp:*</td>
<td>Disabled account</td>
</tr>
<tr>
<td>:/appln</td>
<td>Default login directory</td>
</tr>
<tr>
<td>/bin/sh</td>
<td>&quot;shell&quot; program defining the user environment (optional)</td>
</tr>
</tbody>
</table>

### Table 3.2: Explanation of example /etc/passwd file

Password encryption in the /etc/passwd file is important as UNIX by default allows any system user to read this file. The cat command can be used to display the contents of the passwd file (Garfinkel & Spafford, 1993:23).

Passwords are stored in a one-way encrypted format in the /etc/passwd file. This is done by using the crypt algorithm. The crypt algorithms based on the Data Encryption Standard of the National Institute of Standards and Technology (Garfinkel & Spafford, 1993:30).

When a user attempts a login the /bin/login program does not decrypt the password. Rather it takes the password entered encrypted it and compares the final result to the encrypted password already stored within UNIX.

Password salting can also be used to drastically reduce the effectiveness of password searching and decoding. This mechanism appends a random 12 bit random number (the salt) to the password. The concatenated string is encrypted, and both the 12 bit random number and the encrypted string are stored in the /etc/passwd file. This has the effect of increasing the key search by a factor of 4096 ($2^{12}$) (Garfinkel & Spafford, 1993:30).

The UNIX crypt function takes the user's password as the encryption key and uses it to encrypt a 64-bit block of zeros. The resulting 64-bit block of cipher text is then encrypted again with the user's password. This process is repeated 25 times. The final 64-bits are unpacked into a string of 11 printable characters that are stored in the /etc/passwd file (Garfinkel & Spafford, 1993:30).
The `/etc/passwd` file should be owned by the user `root`. All other users should only have read access to this file.

### 3.3.2.2 Password Conventions

Where passwords are used, UNIX System V enforces the following conventions (Armstrong, et al. 1994:27-32):

- Passwords must consist of a minimum of three characters;
- Passwords must contain at least two alphabetic characters and at least one special caricature from the ASCII table excluding the following:
  - `# @ / ! <backspace> <CTRL>U <CTRL>D`; and
- Passwords are case sensitive.

### 3.3.2.3 Changing Passwords

Passwords can be changed by using the `/bin/passwd` command. The `passwd` command first prompts the user the old password to be re-entered. The `passwd` command then prompts the user to enter the new password. Passwords are not echoed on the screen when typed in (Garfinkel & Spafford, 1993:24). The new password must be entered twice.

The "superuser" can change the password of any user at any time and is not constrained by the password conventions described earlier. A "superuser" can also prevent users from changing their passwords simply by changing the permissions on the `/bin/passwd` program.

### 3.3.2.4 Password Ageing

UNIX System V provides password ageing that can be implemented by the "superuser". The ageing control parameters are appended to each encrypted password in the `/etc/passwd` file, allowing the specification of a minimum and maximum number of weeks before a user can change or are forced to change his/her password (Garfinkel & Spafford, 1993:39).

UNIX System V does not have a password history file but will ensure that new passwords differ with at least three characters from the previous password. By specifying a minimum period that must elapse before a password can be changed prevents users from changing their passwords back to their old password immediately after changing their passwords. Additional password controls can be implemented by using the `/etc/shadow` file (Garfinkel & Spafford, 1993:39).
3.3.2.5 The /etc/shadow File

Since the /etc/passwd file is world readable, as an added feature of security some systems use a shadow file to hold password information. The /etc/shadow file is not readable by ordinary users, therefore an intruder using a computerised search is now denied access to the information needed for password comparison and for identifying log-ins that have not recently being used (Ernest & Young, 1994:36).

The shadow file is not designed to be edited directly, but instead is modified by the passwd command.

If shadow passwords are used care should be taken that all copies of the /etc/passwd file are not maintained on the system where it may be publicly readable.

Example of a shadow file:

```
doug:SpKe$3erWq:7204:18:6
```

Figure 3.2: Example Shadow File

where in the sequence aaaa:bbbbbbbbb:cccc:dd:e

aaaa: is the account name;

bbbbbbbbb: is the encrypted password;

cccc: is the encrypted date of the last password change;

dd: is the expiry time of the password; and

e: is the maximum days for ageing

3.3.3 User names

User names are unique names that identify a user to the UNIX system. User names must be between one and eight characters in length.

3.3.3.1 User Identifiers (UID)

User names identify a user to the UNIX system. UNIX however, internally uses a numeric user-id (uid) and group-id (gid) value to control all access to the file system and system resources. The connection between user names and uid values are stored in the /etc/passwd file (Garfinkel & Spafford, 1993:46).

User identifiers consist of a 16 bit number that the system uses to identify a user. Two users can have the same uid in which case UNIX will view them as the same user no matter if they have different user names or passwords. Two users with the same uid can therefore read and delete each other’s files and kill each other’s programs (Garfinkel & Spafford, 1993:46).
UNIX uses special uids for system accounts. The "superuser" has a uid of "0" while other system accounts such as uucp and daemon have a uid of greater than "0" but less than 100.

As previously mentioned, uids in the /etc/passwd file need not be unique. This can cause a security threat if any UNIX user account has the same uid as root or any other powerful account.

3.3.3.2 Groups and Group Identifiers (GIDS)

Every UNIX user belongs to one or more groups. Like user names and uids, groups have both group names and group identification numbers.

UNIX groups all groups together. The system administrator must assign a user to one or more groups when an account is created for a user. Groups let the system administrator appoint specific groups of users who are allowed to access specific files, directories, or devices (Garfinkel & Spafford, 1993:46).

Each user belongs to a primary group that is stored in the /etc/passwd file. Another file, the /etc/group file contains a list of all the valid user groups on the UNIX system. Users listed in this file are members of these groups listed in this file in addition to their primary group (Garfinkel & Spafford, 1993:46).

An example of the /etc/group file is depicted in figure 3.3:

```
finacc:*:0:root, carel
uucp:*:10:uucp
vision:*:101: clare, caren
finnacc1:*:100
```

Figure 3.3: Example of a /etc/group File

This example can be explained as follows:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>finacc</td>
<td>The group name</td>
</tr>
<tr>
<td>*</td>
<td>The group password described below</td>
</tr>
<tr>
<td>0</td>
<td>The group's GID</td>
</tr>
<tr>
<td>root, carel</td>
<td>The list of users who are in the group</td>
</tr>
</tbody>
</table>

Table 3.3: Explanation of the example /etc/group File
3.3.3.3 Changing Groups, UIDS and GIDS

Under System V UNIX, a user can only reside in one group at a time. With the newgrp command a user can change his/her group. The newgrp command takes a single argument namely the group name to which one wants to change to. If the command succeeds, it spawns a subshell that has a different group ID (Garfinkel & Spafford, 1993:48).

System V allows users to change into a group in which they are not members. However if you want to change into a group in which you are not a member the newgrp command will prompt you for that group's password that is stored in the /etc/group file (Garfinkel & Spafford, 1993:48).

Once uids and gids are assigned they should not be changed without reason as files may be orphaned or reassigned to users.

3.3.4 The UNIX “Superuser”

Every UNIX System comes with a special user in the /etc/passwd file with a UID of “0”. This user is known as the “superuser” and is normally given the name root.

The root account is used by the system itself to accomplish certain tasks such as logging users in and out, recording accounting information, and managing input and output devices. Therefore there is no file that cannot be accessed, written or deleted at any time by the superuser, even security checks can be turned off for any program that is run by the superuser (Garfinkel & Spafford, 1993:50).

The root account should not be used for the personal use of the system administrator although the system administrator will frequently need to become the superuser in order to perform various system administrator tasks.

Often it will be required from computer operations staff to perform administrative functions that can only be performed by the superuser. In these situations it is possible to arrange access permissions to establish privileged user access for these functions without requiring superuser status and capabilities. Refer to “Set-uid (suid) and Set-gid (sgid) Permission in section 3.4.2.3

3.3.4.1 What the Superuser can do

Any process that has an effective UID of “0” runs as the superuser and can perform the following (Garfinkel & Spafford, 1993:50):
• Process Control
  • Change the nice value of any process;
  • Send any signal to a process;
  • Alter the limits for the maximum CPU time as well as maximum file, stack
    segment, and core file sizes;
  • Turn accounting on or off;
  • May be able to bypass login restrictions prior to shutdown; and
  • Become any other user on the system

• Device Control
  • Access any working device at any time;
  • Shut down the computer at any time;
  • Read and modify any memory location; and
  • Set the date and time;

• Network Control
  • Run network services on trusted ports;
  • Put the network in “promiscuous mode” and examine all packets being sent
    on the network; and
  • Reconfigure the network.

• File system Control
  • Run any program at any time;
  • Mount and unmount file systems;
  • Modify user accounts;
  • Enable/Disable quotas and accounting; and
  • Modify any file or program on the system.

3.3.4.2 Becoming the Superuser

To become the superuser the default login root may be used. However, because the normal
security checks and constraints are ignored for the superuser, the use of root cannot be
effectively monitored. Because of this the system administrator and other users should be
equated to login under their individual account and use the /bin/su command to switch
to the root user whenever superuser capability is required (Garfinkel & Spafford, 1993:54).

Typing su with no arguments instructs UNIX that the user wishes to become a superuser.
The system however prompts the user using the su command for the root password before
allowing the user to become a superuser.
All unsuccessful su attempts are logged in the /usr/adm/message file. Some UNIX systems maintain an audit trail of all superuser logins done by using the su command, both successful and unsuccessful attempts are stored in the sulog file (Garfinkel & Spafford, 1993:55).

3.3.4.3 $HOME Directory

The $HOME directory is the directory that becomes the user’s current directory following a login. This directory is specified in the /etc/passwd file. This directory also contains certain files, including .login and .profile that are executed automatically when a user logs into the system. These programs are executed by the user’s shell and establish the general working environment for the session, including the PATH variable and calling application menus for users who are not permitted access to the UNIX operating system prompt.

UNIX users who have access to the operating system through one of UNIX shell programs must use the cd or chdir command to change their own working directory to another directory on the system. Changing the current working directory is subject to directory access permissions.

Access to the /bin/cd command should be restricted to limit movement within the system. A user can however still be able to execute programs not in his/her working directory by entering the full path name. (Refer 3.3.4.4)

3.3.4.4 The UNIX Path

The PATH = entry in the /etc/profile file establishes a system-wide path that is used by UNIX to locate programs that do not exist in a user’s current working directory.

Users can however bypass the UNIX PATH by entering the full system path in addition to a command. Individual users PATH values are specified in the $HOME/.profile file. A user’s PATH value overrides the UNIX system PATH value. Users should therefore be restricted to using the restricted shell rsh that does not allow users to change their .profile file or write access to the .profile file should be removed (Garfinkel & Spafford, 1993:143).

The PATH variable for the superuser should never contain a “.” entry, as this represents the current directory value and means that where a command is executed from within a UNIX shell, the current directory will be searched first.

3.3.5 UNIX SHELLS

UNIX uses shells to read commands entered by the user and to run other programs. The initial login process for each user ends normally in either one of UNIX’s shells or an application software front-end program. Shells and programs to be loaded when a login is performed are specified in the etc/passwd file (Ernest & Young, 1994:18).
The UNIX shell runs other programs by first executing one of the fork family of instructions to create a second process. The second process then uses one of the exec family of instructions to run a new program, while the first process waits for the second process to finish.

### 3.3.5.1 The Restricted Shell

UNIX uses another command interpreter called rsh to minimise the dangers posed by open accounts. When rsh starts up it executes the commands in the $HOME/.profile file and the following restrictions take effect (Garfinkel & Spafford, 1993:23):

- The user cannot change the current directory;
- The user cannot change the value of the PATH statement;
- The user cannot execute commands containing slashes; and
- The user cannot redirect output with > or >>.

RSH will immediately exit if a user tries to interrupt it while it is busy processing the $HOME/.profile file.

### 3.3.6 UNIX Log Files

UNIX maintains a number of log files that will record when users log in or out of the system and log the commands used by the users during that session. This file forms part of UNIX's auditing system.

#### 3.3.6.1 The Logging of UNIX Commands

UNIX is able to log every command issued by system users, but however is not able to log arguments used in the command, nor the directory in which that command was used.

The logging of commands issued by users is performed by the kernel. Every time a process ends, the kernel writes a 32 bit record to the /usr/adm/acct file that includes (Garfinkel & Spafford, 1993:125-130):

- The name of the user that issued the command;
- The name of the command issued;
- The amount of CPU time used;
- The time the process was exited; and
- Flags that include the following:
  - X: if the command was terminated by a signal;
  - S: if the command was issued by the superuser;
  - F: if the command ran after a fork, but without an exec; and
  - D: if the command generated a core file when it exited.
3.3.6.2 The Logging of User Logins and Logouts

UNIX System V contains the following standard log files as set out in table 3.4 (Garfinkel & Spafford, 1993:125):

<table>
<thead>
<tr>
<th>FILE NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/adm/lastlog</td>
<td>This file is used to log the user’s most recent login time.</td>
</tr>
<tr>
<td>/etc/utmp</td>
<td>This file logs all users logged in on the system and maintains a list of all users currently logged into the system.</td>
</tr>
<tr>
<td>/etc/wtmp</td>
<td>This file logs a record every time a user logged into or out of the system</td>
</tr>
</tbody>
</table>

Table 3.4: UNIX Log Files

The entries in the utmp and wtmp file contain the following information (Garfinkel & Spafford, 1993:126):
- The user name;
- The terminal line number;
- The process ID;
- A code for the entry;
- The exit status of the process; and
- The time that the entry was made.

Programs such as who, users and finger that reports the users that are currently logged into the system do so by scanning the /etc/utmp file.

3.4. THE UNIX FILE SYSTEM

The UNIX file system controls the way that information in files and directories is laid out on the disk. It controls which users can access what and how they can access this information. The file system therefore constitutes the basic tool for enforcing UNIX security (Armstrong, et al, 1994:8).

There are three types of UNIX files namely:
- Regular files that hold executable programs and data;
- Directories that contains other files and sub-directories; and
- Device files that refer to computer hardware such as memory, tape device disk drives, etc.
3.4.1 UNIX File Storage and Inodes

UNIX stores all files in a tree-structured file system built from files and directories. Individual files are identified by unique file numbers called inode numbers. Inodes are assigned to a file by the UNIX system on creation of the file. UNIX uses this inode number to control and locate files on the system (Armstrong, et al., 1994:8).

The inode stores the following general information on each file (Ernest & Young, 1994:39):
- The location of the file on the disk;
- The type of file (directory, ordinary etc.);
- The file size;
- Protection bits;
- The physical address on the disk of the first block of the file;
- The time the file’s inode was last modified (ctime);
- The time the file’s contents were last modified (mtime);
- The time the file was last accessed (atime); and
- A reference count of the number of links the file has.

The following security related information for each file is also stored:
- The file’s mode bits (also referred to as file permission or permission bits);
- The file’s owner (a uid); and
- The file’s group owner (a gid).

3.4.2 UNIX File permissions

Three groups of three characters that form nine permission bits are associated with each UNIX file. These permission bits describe the actions that the file’s owner, group members and other users on the system can perform with a file.

UNIX file permissions can best be described by the following example as set out in figure 3.5 on the following page:
Figure 3.5: *UNIX File Permissions*

The first character of the file’s permission bit indicates the type of file. Valid first characters are:
- `-` The file is a plain file;
- `d` The file is a directory;
- `c` The file is a character device (tty or printer);
- `b` The file is a block device (disk or tape); and
- `=` The file is a FIFO.

The remaining characters of the file’s permissions define the access rights of users to the file. The valid types of permission are:
- `r` *(r)ead* allows users only to open and read the contents of a file;
- `w` *(w)rite* allows users to overwrite the file or modify its contents;
- `x` *(x)ecute* allows users to run the file by entering its path name; and
- `s` *(SUID)* allows a user to execute the program using the access rights of the owner of the program.

The nine characters or read, write and execute permissions are arranged in blocks of three according to the following respective user group structure:
- *owner:* The file’s owner;
- *group:* User’s who are in the file’s group; and
- *other:* Everyone else on the system except the superuser.

---

**Table:**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Owner</th>
<th>Group</th>
<th>Others</th>
<th>Date</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>prw-r-</td>
<td>peter</td>
<td>finoc</td>
<td>Feb 20</td>
<td>12:27</td>
<td>pipefile</td>
<td></td>
</tr>
<tr>
<td>-wxs-x-</td>
<td>root</td>
<td>sys</td>
<td>Apr 15</td>
<td>14:10</td>
<td>setuidfile</td>
<td></td>
</tr>
<tr>
<td>-rw-r-x-</td>
<td>clare</td>
<td>other</td>
<td>May 12</td>
<td>16:15</td>
<td>setgidfile</td>
<td></td>
</tr>
<tr>
<td>rw-</td>
<td>peter</td>
<td>other</td>
<td>May 13</td>
<td>10:40</td>
<td>setuidnonsx</td>
<td></td>
</tr>
<tr>
<td>-rw-r-x-</td>
<td>john</td>
<td>other</td>
<td>Jun 1</td>
<td>12:43</td>
<td>lockedfile</td>
<td></td>
</tr>
<tr>
<td>rwx-</td>
<td>clare</td>
<td>other</td>
<td>Jul 19</td>
<td>20:20</td>
<td>stickybitfile</td>
<td></td>
</tr>
</tbody>
</table>
Therefore in the example:

```
rwxr-xr-x
```

the file permissions are interpreted as follows:

<table>
<thead>
<tr>
<th>File</th>
<th>owner</th>
<th>group</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>rwx</td>
<td>r-x</td>
<td>r-x</td>
</tr>
</tbody>
</table>

**Figure 3.6: Example of File permissions**

Where the file is an ordinary file, the file's owner has got read, write and execute permission. The file's group has only read and execute permissions while all other users also have read and execute permission.

A user can have only execute access without read access. This allows a user to run the program without reading its contents and is often used to hide certain functions of the program.

By using the `chmod` command a user can change a file's permission at any time if the user is the owner of that file.

### 3.4.2.1 UMASK

Umask is a three digit octal number that specifies default access permissions to newly created files. Umask operates by specifying the file permissions that are not to be given to the newly created files.

On most UNIX systems, files are created with default permission of '666' which means that all users have read and write access. Executable files and other files created with a default permission of '777' and a umask value of '022' would deny write access to either type of file immediately after creation. That is.

```
666  Default file permission on creation of the file

-022 Less: The umask value

644  Actual file permission granted to a new file
```

A umask with the value of "000" therefore forms an insecure mask, as this does not deny any permissions for the file created. A umask with a value of "077" will deny all access to group and others when a file is created.
Umask is set in the /etc/profile file or in individual users .login, .cshrc or .profile files.

3.4.2.2 Directory Permissions

UNIX uses ordinary files to store the contents of directories, and as such the permission bits pertaining to files also apply to directories. However UNIX treats directories in a different way than ordinary files and therefore the meaning of each permission bit has a slightly modified meaning.

- **r** (r)ead allows the user to find out which files are in the directory but cannot produce long file listings;
- **w** (w)rite allows the user to add or remove files or links in the directory; and
- **x** e(x)ecute allows user to find out more detailed information about the files in the directory such as the each file’s owner and the length of the file. The user can also make this directory its current directory and is able to open files inside this directory.

Therefore a user must have execute rights to a directory to make that directory his/her current working directory. A user without execute rights to a directory cannot access files in that directory, even if he is the owner of those files. A user without read access rights but with execute rights to a directory cannot list the files in that directory. However if they have access to individual files they can run programs or open files in that directory. A user must have write and execute rights to a directory to delete files in that directory.

3.4.2.3 Set-uid(suid) and Set-gid(sgid) Permissions

The suid and sgid functions allow unprivileged users to perform privileged functions and commands by granting privileges to the command rather than the user.

When the suid program is executed, its effective uid becomes that of the user who created the program, rather than the user who is running the program. Therefore suid programs will execute with the UNIX access privileges of the owner and not the user who started the program. Such files are particularly dangerous if owned by root given that these programs will execute with root privileges regardless of the user executing it.

When the sgid program is executed, its effective gid becomes that of the owner’s group who created the program, rather than the group who is running the program. Therefore sgid programs will execute with the UNIX access privileges of the owner’s group and not the group who started the program.

Any files with the suid/sgid bits turned on, that is writeable by users other than the owner of the file are dangerous as other users are able to overwrite these files with their own executable code and gain additional access privileges.
3.4.3 Encryption

UNIX comes with a crypt command that allows users to encrypt files. When files are sent over the network they should be encrypted and compressed with the pack command.

It should however be noted that the crypt command has been broken and does not have much benefit as an encryption tool. The DES file encryption tools namely encrypt and decrypt should be used instead.

3.5 SHELLS

Most users should not be granted access to shells as shells give users an opportunity to use powerful utilities and scripts through which users can basically write their own programs.

UNIX is shipped with basically three shells namely:
- The Borne shell;
- The C shell; and
- The Korn shell.

Both the Borne and C shells have restricted versions that restrict users' movement and ability to execute programs through the UNIX operating system.

Shells should only be used for more sophisticated users if necessary, as users have to be familiar with UNIX commands to work with shells.

3.6 UTILITIES AND APPLICATIONS

From a user's start-up program or process, as defined in the /etc/passwd file, a user can invoke an application or a utility.

Utilities will be mainly used by IS staff in the performance of their daily tasks.

Normally users should only access applications by a menu type application that allows them to start their various applications, perform their work, and return to the menu. Menu application should not allow access to shells. The termination of the menu application should result in the user logging off the system.

Multi-user applications for example databases and accounting packages normally contain their own security features. The security features of these applications typically classify users into classes and determine the data and functions that a user can use. Security features for these packages normally specify the type of access permitted within each module. In such situations the UNIX file access permissions are normally set when the program is installed.
3.7 THE UNIX SYSTEM

The UNIX system consists of the rest of the UNIX operating system that allows the system to function properly.

3.7.1 The Kernel

The kernel is mainly responsible for the managing of processes running on the system and also manages the interaction of processes with hardware.

3.7.2 UNIX Processes

All activity on the UNIX system is performed by processes. Every process is assigned a unique number called the process identifier, or PID. All processes running on the system can be listed by using the ps(1) command.

Every process running on the system has two user identifiers namely:

- A real UID (RUID). The ruid is the actual uid if the person running the program. It is normally the same uid as that of the person who logged into the computer; and
- The effective UID (EUID). The euid identifies the actual privileges of the process that is running.

3.7.2.1 Process Priority and Niceness

Although UNIX is a multi-tasking operating system it still can only run one process at a time. UNIX switches between different processes every fraction of a second so that each process can be worked on. A small part of the kernel called the process scheduler decides which process is allowed to run at any given moment and how much CPU time that process gets.

The process scheduler computes a priority for every process in order to decide which process gets to run next. The process with the lowest priority number gets to run first. A process’s priority is determined by a complex formula that takes into account what the process is doing and how much CPU time the process has already used. A special number called the nice number biases this calculation in the sense that the lower the nice number of a process the higher its priority.

On most UNIX system the nice number is limited from -20 to 20. Most processes have a nice number of “0”. With the program /etc/renice a process’s nice number can be changed. Normal users however are only allowed to increase the nice numbers of their own processes. Only the superuser can lower a nice number or raise the nice number of other user’s processes.

If an intruder is caught on a UNIX system the damage done by the intruder can be lessened by increasing the nice numbers of the intruder’s processes.
3.7.2.2 UNIX Signals

Signals are the mechanism through which UNIX control processes. A signal is a 5 bit message to a process that requires immediate attention. Each signal has associated with it a default action. Signals are generated by exceptions such as:

- Attempt to use illegal instructions;
- Certain mathematical operations;
- Another program using the kill command; and
- The user interrupting a process by pressing the interrupt key.

Signals are normally used between processes for process control and within processes to indicate special circumstances that should be handled immediately.

3.7.2.3 The Kill Command

The owners of a process or the superuser can terminate a process by using the kill command. The kill command works by sending a signal to the process that the user wants to terminate. Killing certain key processes may cause the UNIX system to halt abnormally causing the corruption and loss of data.

3.7.3 Daemons

Daemons are processes that run in memory the whole time performing certain functions to ensure that the system is running smoothly. Daemons also have a PID number and are therefore susceptible to manipulation.

3.8 CONCLUSION

Each of the layers/components in the predefined environment that will have an effect on the control objectives have been identified. The components identified for the different layers in the UNIX access path will be used to develop an audit questionnaire that can be used to evaluate UNIX security. This questionnaire will be the subject of Chapter 4 and will also indicate what control objectives will be complied with.
# CHAPTER 4

DEVELOPMENT OF AN AUDIT PROGRAMME FOR THE SELECTED UNIX ENVIRONMENT

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 INTRODUCTION</td>
<td>58</td>
</tr>
<tr>
<td>4.2 PREPARATION</td>
<td>58</td>
</tr>
<tr>
<td>4.3 DEVELOPMENT OF THE AUDIT PROGRAMME</td>
<td>59</td>
</tr>
<tr>
<td>4.4 AUDIT PROGRAMME</td>
<td>60</td>
</tr>
<tr>
<td>4.5 CONCLUSION</td>
<td>71</td>
</tr>
</tbody>
</table>
4.1 INTRODUCTION

In this chapter an audit programme will be developed for the different components of UNIX, some of which have been described in chapters 2 & 3.

The procedures described in section 4.4 below are neither complete, nor can all the steps be taken on a given system. Selecting procedures to be performed is a matter of judgement after assessing the particular risks and after gaining an understanding of the system lay-out and components' set-up.

It is not sufficient to work through this programme alone, as many features of the UNIX operating system can and will change over time. The auditor should also read through user and operating guides that set out control and security guidelines before commencing with the installation review.

General controls over the authorisation, testing and approval of software changes have not been included in the audit programme as they are of a general nature and are relevant in most computer environments. Only those controls that might have an effect on program change controls will be included in the audit programme.

4.2 PREPARATION

Before starting a UNIX installation review the following must be done:

- Identify and discuss the organisational structure;
- Identify and discuss the organisation of the EDP department;
- Details of the equipment and significant applications that are in use should be obtained;
- Diagrams setting out the various components of the network, including their location and method of communication should be drawn up. These diagrams should also show the extent of use of the both the hardware and software platforms;
- A list of all user profiles defined to the system should be obtained. This list should include the name of the person, job title and department;
A list of user groupings, their members and their purpose; and
Copies of the following documents should also be obtained:
- Disaster Recovery Plan;
- Security Policies and Procedures;
- Change Control procedures.

4.3 Development of the Audit Programme

The audit programme was developed by identifying the different layers in the UNIX operating system. The different layers of the UNIX operating system was identified by applying the access path model to the predefined environment.

By applying this model to the predefined environment it was possible to identify the different layers of the UNIX operating system through which a user “moves” when accessing data, files or programs. In each layer of the UNIX operating system different components, that have an effect on computer security, have been identified.

The questionnaire developed in 4.4 is a result of:

- The evaluation of the key components identified in Chapter 2 and 3 in each layer and its effect on computer security;
- The evaluation of the parameters in each of these components that will have an effect on computer security. The discussion of all these parameters in detail falls outside the scope of this research essay;
- The evaluation and inclusion of well known and widely published security weaknesses of UNIX;
- The evaluation of numerous security questionnaires that are aimed at system administrators and users;
- The evaluation of well known and accepted general controls; and
- The evaluation of UNIX advisories on computer security.

An evaluation of the effect of the above on the control objectives set in Chapter 1 of this essay was done. The objectives at which each question is aimed and that will be met if the installation under review complies with properly, are indicated on the audit programme.
<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Chap. Ref.</th>
<th>Compl</th>
<th>Valid</th>
<th>Acc</th>
<th>Cont</th>
<th>Int</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NETWORK CONTROLS</td>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Trusted Hosts</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>Are trusted hosts used on the system and if so why?</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>List the trusted hosts and confirm that only a small number of hosts are used. Compare the list of trusted hosts with the approved list of the system administrator</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>Are all trusted hosts within the organisations' domain and under its control? If trusted hosts are outside the domain of the organisation arrange for a review of the security of that system or obtain a report from that organisation's independent auditors in connection with the security of that system.</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>Are netgroups being used for easier management (when NIS is used)?</td>
<td>3.2.11</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>Confirm that fully qualified host names are used</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>Confirm that there is no &quot;+&quot; sign by itself in the ( /etc/hosts.equiv ) file.</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>Confirm that the first character of the ( /etc/hosts.equiv ) file is not a &quot;-&quot;.</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>h</td>
<td>Confirm that the ( /etc/hosts.equiv ) file's permission is set to 600.</td>
<td>3.4.2</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>If trusted hosts are not used confirm that the ( /etc/hosts.equiv ) file does not exist.</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>Confirm that the owner of the ( /etc/hosts.equiv ) file is root.</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>Confirm that no user has a ( .rhosts ) file in their directory</td>
<td>3.2.7</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>If used obtain reasons for the use of ( $HOME/rhosts ) files.</td>
<td>3.3.4.3</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>Confirm that the first character of the ( $HOME/rhosts ) is not a &quot;-&quot;.</td>
<td>3.3.4.3</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>Confirm that the ( $HOME/rhosts ) file's permission is set to 600.</td>
<td>3.4.2</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>Confirm that the owner of the ( $HOME/rhosts ) file is also the owner of the account.</td>
<td>3.3.4.3</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>Confirm that the ( $HOME/rhosts ) file does not contain a &quot;+&quot; on any line.</td>
<td>3.3.4.3</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>NFS</td>
<td>3.2.12</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>Confirm that NFS is disabled if not in use by the organisation</td>
<td>3.2.12</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
b. Confirm that **NFS** traffic is filtered at the router (Filter TCP/UDP on port 111 and on port 2049) to prevent computers not on the organisation's subnet from accessing file systems exported by your system.

c. Confirm that **NFS** port monitoring is enabled to ensure that all calls to mount a file system will only be accepted from port < 1024 (root access).

d. Confirm that `/etc/exports` is only used for files that need to be exported.

e. Confirm that the following option -access=host.domain.au is used in `/etc/exports` to prevent file systems from being exported to the world.

f. Confirm that file systems that are exported is set to read-only through enquiry from the system administrator.

g. Confirm that the file permissions for `/etc/exports` are set to 644.

h. Confirm that `/etc/exports` is owned by root.

i. Review the `/etc/exports` file and identify all file systems that do not have a host associated with them. These file systems are mountable by all hosts and should therefore only be for public use.

j. Review documents that authorise the mounting of specific directories or file systems by specified hosts.

k. List the system **NFS** servers that allow root on clients to access their resources as if they are root on the server.

l. Check the mode of the `netgroup` file.

m. Confirm the aptness of the entries in the `netgroup` file.

n. Run the `showmount (dfmounts)` with the `-a` option to list all the currently active mounted **NFS** file systems and follow up any unusual directories, etc., that are mounted.

o. Confirm that when file systems are mounted the `nosuid` option is used to prevent the unauthorised use of `suid` and `sgid` programs.

1.3 **Finger**

a. Confirm that the finger program has been disabled. If the program is in use obtain reasons why.

b. If `finger` is used confirm that `fingerd` is produced later than 5 Nov. 1988.

1.4 **The `/etc/hosts.lpd` File**

a. Confirm that the first character in the `/etc/hosts.lpd` is not a '-'.

b. Confirm that the permission of the file `/etc/hosts.lpd` is set to 600.

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>CHAP Ref.</th>
<th>Compl</th>
<th>Valid</th>
<th>Acc</th>
<th>Cont</th>
<th>Int</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Confirm that <strong>NFS</strong> traffic is filtered at the router (Filter TCP/UDP on port 111 and on port 2049) to prevent computers not on the organisation’s subnet from accessing file systems exported by your system.</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>c</td>
<td>Confirm that <strong>NFS</strong> port monitoring is enabled to ensure that all calls to mount a file system will only be accepted from port &lt; 1024 (root access).</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>d</td>
<td>Confirm that <code>/etc/exports</code> is only used for files that need to be exported.</td>
<td>3.2.12</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>e</td>
<td>Confirm that the following option -access=host.domain.au is used in <code>/etc/exports</code> to prevent file systems from being exported to the world.</td>
<td>3.2.12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>f</td>
<td>Confirm that file systems that are exported is set to read-only through enquiry from the system administrator.</td>
<td>3.2.12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>g</td>
<td>Confirm that the file permissions for <code>/etc/exports</code> are set to 644.</td>
<td>3.4.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>h</td>
<td>Confirm that <code>/etc/exports</code> is owned by root.</td>
<td>3.2.12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>i</td>
<td>Review the <code>/etc/exports</code> file and identify all file systems that do not have a host associated with them. These file systems are mountable by all hosts and should therefore only be for public use.</td>
<td>3.2.12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>j</td>
<td>Review documents that authorise the mounting of specific directories or file systems by specified hosts.</td>
<td>3.2.12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>k</td>
<td>List the system <strong>NFS</strong> servers that allow root on clients to access their resources as if they are root on the server.</td>
<td>3.2.12</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>l</td>
<td>Check the mode of the <code>netgroup</code> file.</td>
<td>3.2.11</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>m</td>
<td>Confirm the aptness of the entries in the <code>netgroup</code> file.</td>
<td>3.2.11</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>n</td>
<td>Run the <code>showmount (dfmounts)</code> with the <code>-a</code> option to list all the currently active mounted <strong>NFS</strong> file systems and follow up any unusual directories, etc., that are mounted.</td>
<td>3.2.12</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>o</td>
<td>Confirm that when file systems are mounted the <code>nosuid</code> option is used to prevent the unauthorised use of <code>suid</code> and <code>sgid</code> programs.</td>
<td>3.2.12</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>1.3</td>
<td><strong>Finger</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>a</td>
<td>Confirm that the finger program has been disabled. If the program is in use obtain reasons why.</td>
<td>3.2.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>b</td>
<td>If <code>finger</code> is used confirm that <code>fingerd</code> is produced later than 5 Nov. 1988.</td>
<td>3.2.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>1.4</td>
<td><strong>The <code>/etc/hosts.lpd</code> File</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>a</td>
<td>Confirm that the first character in the <code>/etc/hosts.lpd</code> is not a <code>'-'</code>.</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>b</td>
<td>Confirm that the permission of the file <code>/etc/hosts.lpd</code> is set to 600.</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>No</td>
<td>Description</td>
<td>CHAP Ref</td>
<td>Compl</td>
<td>Valid</td>
<td>Acc</td>
<td>Cont</td>
<td>Int</td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>1.5</td>
<td>The /etc/inetd.conf File</td>
<td>3.2.3</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>a</td>
<td>Confirm that permissions to the /etc/inetd.conf file is set to 644.</td>
<td>3.4.2</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>Confirm that the owner of the /etc/inetd.conf file is root.</td>
<td>3.2.3</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>Confirm that all services that are not required are disabled for example finger and tftp.</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>Review the /etc/inetd.conf file for any daemons that normally should be loaded but that has been disabled.</td>
<td>3.2.3</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>FTP</td>
<td>3.2.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>If FTP or anonymous FTP is used determine the validity thereof.</td>
<td>3.2.9.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>Confirm that the most recent version of FTP daemon are in use.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>Confirm that the FTP server does not have the SITE EXEC command by telnetting to port 21 and typing SITE EXEC. If SITE EXEC is present ensure that the most recent version is in use because older versions allowed anyone to gain shell through port 21.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>Confirm that any commands from “ftp/bin, “ftp/sbin or similar directories that can be executed from SITE EXEC do not contain system commands or include a shell as this can allow users to gain root access.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>Confirm that the use of FTP has been restricted to a separate server to localise the potential risks of using FTP.</td>
<td>3.2.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>Confirm that all delete, overwrite, rename, chmod and umask options are not allowed for guests and anonymous users in configurable FTP.</td>
<td>3.2.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>Confirm that a /etc/ftpusers file has been set up to deny certain users the use of FTP. At a bare minimum this should include root, bin, uucp, ingres, daemon, news, nobody and all vendor supplied accounts.</td>
<td>3.2.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>Ensure that an invalid password and user shell is used for the FTP entry in the system passwd file.</td>
<td>3.2.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>Confirm that a copy of the real /etc/passwd file do not exist as “ftp/etc/passwd.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>Confirm that a copy of the real /etc/group file do not exist as “/etc/group.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>Confirm that the length of the files “ftp/.rhosts and “ftp/.forward are zero and that their permission are set to 600 with root the owner of this file.</td>
<td>3.4.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Description</td>
<td>Chap Ref</td>
<td>Compl</td>
<td>Valid</td>
<td>Acc</td>
<td>Cont</td>
<td>Int</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>i</td>
<td>Confirm that specified directories have been assigned for files being sent and received by FTP.</td>
<td>3.2.9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>m</td>
<td>Enquire whether anonymous FTP allows files to be read and written to the system. If so document and evaluate procedures that are used to review this files prior to their transmittal and who are responsible for the transmission of these files.</td>
<td>3.2.9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>n</td>
<td>Confirm that no files are owned by the FTP account or have the same group as the FTP account.</td>
<td>3.2.9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>o</td>
<td>Confirm that anonymous FTP users cannot create files or directories in any directory.</td>
<td>3.2.10</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>p</td>
<td>Confirm that FTP users have only read access writes to directories and file in public areas.</td>
<td>3.2.9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>q</td>
<td>Confirm that the permissions of the FTP home directory (~/ftp) are set to 555 and that the owner is root.</td>
<td>3.4.2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>r</td>
<td>Confirm that the system sub directories (~/ftp/etc and ~/ftp/bin) have permissions of 111 only and that the owner of these directories are root.</td>
<td>3.4.2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>s</td>
<td>Confirm that the ~/ftp/etc/* have permissions set to 444 and that root is the owner of these files.</td>
<td>3.4.2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>t</td>
<td>Confirm that /usr/spool/mail/ftp is owned by root and that permissions are set to 400.</td>
<td>3.4.2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>u</td>
<td>Ensure that writeable directories are limited and that these directories if they exist are not readable, is owned by root and have permissions set to 1733.</td>
<td>3.4.2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>v</td>
<td>Confirm that disks mounted from other machines to the ftp hierarchy are only done if they are read-only.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>w</td>
<td>Document and evaluate procedures in place for the transfer of files received through FTP to the production system.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**1.7 Aliases**

| a  | Confirm that the "decode" alias has been removed from the /etc/aliases file. | 3.2.4    | ✓     | ✓     | ✓   | ✓    | ✓   |
| b  | Confirm that all programs executable by an alias are owned by root, have permission of 755 and are stored in a systems directory. | 3.4.2    | ✓     | ✓     | ✓   | ✓    | ✓   |

**1.8 Sendmail**

<p>| a  | Confirm that the latest version of sendmail is in use. |          | ✓     | ✓     | ✓   | ✓    | ✓   |
| b  | Confirm that the line in the /etc/sendmail.cf file starting with &quot;OW&quot; only has &quot;**&quot; next to it. |          | ✓     | ✓     | ✓   | ✓    | ✓   |</p>
<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
</tr>
</thead>
</table>
| c  | Use the following tests to verify whether sendmail is outdated and susceptible to known security vulnerabilities:  
  % telnet host name 25  
  wiz  
  debug  
  kill  
  quit  
  %  
  If the response from the system is not "500 Command unrecognised" after each of the commands entered, then the version of sendmail is vulnerable and should be updated. |
| d  | Confirm that sendmail (8)'s logging is increased to a minimum log level of 9. (This help to detect attempted exploitation of the sendmail (8) vulnerabilities). |
| e  | Confirm that a minimum level of "info" is set for mail messages to be logged to the console and/or the syslog file. |
| f  | Ensure that the "wizard" password is disabled in the sendmail.cf file. |
| a  | Confirm that UUCP and the shell that it executes for logging in are disabled, if it is not used by the organisation. |
| b  | Confirm that all .rhosts are removed from UUCP's home directory. |
| c  | Confirm that the file L.cmds is owned by root. |
| d  | Confirm that no UUCP owned file are world writable. |
| e  | Confirm that different UUCP logins have been assigned for each site that needs UUCP access. |
| f  | Ensure that commands that can be executed with each UUCP login have been limited to the bare minimum. |
| g  | Enquire if sequence checks are performed before each login. |
| h  | Enquire if UUCP logging are enabled and if so what information are logged and who is responsible for following up the log? |
| i  | Confirm that UUCP avoids trusting remotes purely on unauthenticated system names. |
| j  | Inspect UUCP directories and file and evaluate the sensibility of modes assigned there to |
| k  | Confirm propriety of local file names that remote systems can send to this system or receive from it. |
| l  | Confirm propriety of remote command requests that uuxqt will accept to run locally. |
Review the /usr/lib/uucp/L.sys file and agree the listed names with the authorised list of UUCP links and telephone numbers. Ensure that each user has a password.

Review the /usr/lib/uucp/L.cmds file and ensure that programs listed in this file are only those programs that the outside UUCP user needs to perform the task assigned to it.

Review the contents of the /usr/lib/uucp/USERFILE as follows:
- For line beginning with a comma review the directory to which these users have access. This directory should be the lowest level of a directory tree and ensure that both read and write directories are appropriate.
- For sites where a dial back facility exists ensure that a "c" exists after the system and/or user name.

Confirm that the /usr/lib/uucp/L.sys file is owned by the user uucp and have a permission of 600.

MODEMS
a. Confirm that modems are configured as dial back modems where possible.

b. Confirm that for line or modem failures sessions are terminated and the user logged out.

c. Enquire if separate lines than normal telephone lines are used for modems.

Confirm that the reprogramming of modems are only done by the system administrator and not by any users.

e. Review and evaluate the file permissions of the modem configuration and control files to determine who is allowed to make changes to these files and who is the owner of these files.

f. Review modem monitoring activities, and any evidence of such review procedures. Evaluate actions taken during these procedures by the client.

g. Where dial back modems are in use verify a selection of telephone numbers.

R-Commands
a. Confirm that all r commands for example rlogin, rsh, etc. are disabled if not needed.

b. If the r commands are being used confirm that ports 512, 513 and 514 are filtered at the router. This will stop users outside the organisation's domain from exploiting the r commands.

PASSWORDS AND ACCOUNTS

2.1 Policies

a. Evaluate the contents of the document setting out the password policies of the organisation.
2.2 **User Profiles**

a. Document and evaluate the following:
   - Who and how are user profiles determined;
   - If group or shared profiles are used and if so the reason therefor;
   - If any guest accounts are active on the system and if so document the controls in place to restrict the use of these accounts.
   - The standard settings that are used when a user account is defined.

b. Document the shared accounts that exist and evaluate the controls in place to minimise their abuse.

c. Scrutinise the `/etc/passwd` file to determine if every user on the system has a unique UID number.

d. Inquire if the standard user profiles installed with the system are still required.

e. Review and evaluate the procedures in place to create and delete users.

f. Review documentation that pertains to the creation or deletion of users.

g. Select a sample of user profile names from the `/etc/passwd` file and compare this to the list of personnel that are authorised to have access on the system.

h. Obtain and review the list of groups and group membership for reasonableness.

i. Search for users that can change group membership and review this for reasonableness.

j. Enquire if every user has his/her own home directories in which personal files can be stored.

k. Evaluate the permission bits on the home directories of a sample of users to determine if only the specific user has access to this directory or not. If other users do have access to these directories review and evaluate the aptness thereof.

l. Evaluate the permission bits on the user's individual configuration files to determine if access rights to these files are appropriate.

m. Confirm that the configuration file has a umask value of 077.

2.3 **Passwords, User Accounts and Log-ins**

a. Enquire if users have been informed of effective password selection and the importance thereof.
<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>CHAP Ref.</th>
<th>Compl</th>
<th>Valid</th>
<th>Acc</th>
<th>Cont</th>
<th>Int</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Confirm that minimum password lengths are enforced.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>c</td>
<td>Confirm that password ageing is implemented by scrutinising the /etc/passwd file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>d</td>
<td>Confirm whether password expire warnings are enforced.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>e</td>
<td>Determine if password exception lists are kept.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>f</td>
<td>Determine if login delay penalties are enforced on failed logins.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>g</td>
<td>Establish whether rules and policies are laid down for operating as the superuser. If so evaluate these rules and policies and scrutinise any evidence of rule and policies being enforced.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>h</td>
<td>Confirm that the root password is only known to those users that really needs it.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>i</td>
<td>Confirm that all users knowing the root password belongs to the same group with a group id of &quot;0&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>j</td>
<td>Enquire whether the whole path is entered when commands are issued by the superuser to limit the possibility of accidentally running a trojan horse.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>k</td>
<td>Review and evaluate the su log for evidence of users switching to root.</td>
<td>3.3.4.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>l</td>
<td>Inquire which users and groups belongs to root and evaluate the appropriateness thereof</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>m</td>
<td>Confirm that root does not have a &quot;./rhosts&quot; file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>n</td>
<td>Confirm that &quot;.&quot; is not in root’s search path.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>o</td>
<td>Confirm that both root login and cron job files do not source any other files not owned by root or that are group or world writeable.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>p</td>
<td>Enquire whether when user login to root they do so by first logging in under their own account and the su to root.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>q</td>
<td>Review and evaluate the follow up procedures and action taken by the system administrator on the su log. Scan the /usr/adm/messages for any bad su attempts.</td>
<td>3.3.4.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>r</td>
<td>Review, document and evaluate the controls pertaining to the root password.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>s</td>
<td>Review, document and evaluate controls that are in place over the root password in case of it being used in an emergency by users who do not normally have access to this password. (Document what procedure should be followed to obtain the password and how it is then subsequently changed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>t</td>
<td>Confirm that the permission on the root.profile file is set to 077.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>u</td>
<td>Confirm that all .profile files have permissions set to read and write only by the user in which login directory they reside.</td>
<td>3.4.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>No</td>
<td>Description</td>
<td>CHAP Ref.</td>
<td>Compl</td>
<td>Valid</td>
<td>Acc</td>
<td>Cont</td>
<td>Int</td>
</tr>
<tr>
<td>----</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>v</td>
<td>By using the find command extract a list of all the files and directories that are owned by root but whose group ownership is not that of a group to which root belongs and/or is not root’s default group. Discuss these files with the system administrator to determine the appropriateness of such usage.</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>Confirm that NIS is disabled if not needed.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>Confirm that the only computers that have a “+” entry in the /etc/passwd file are NIS clients. The NIS master server should not have a “+” sign in the above mentioned file.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>Confirm that the /etc/rc.local is set up to start ypbind with the -s option.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
| z  | Establish whether a password shadow file is being used and if so:  
  - determine and document the name and path of the file;  
  - confirm that read access to the file is prohibited;  
  - confirm that read access to the file is prohibited.                                                                                                                                                                                                                                                                                                          |           |       | ✓     | ✓   | ✓    |     |
| aa | Confirm that where password shadow files are not used the /etc/passwd file cannot be read anonymously by UUCP or TFTP.                                                                                                                                                                                                                                                                                                                      |           |       | ✓     | ✓   | ✓    |     |
| bb | Enquire whether the algorithm crypt(3) have been altered to increase the number of encryption rounds.                                                                                                                                                                                                                                                                                                                                                      |           | ✓     | ✓     |     |      |     |
| cc | Ensure that user areas are adequately backed up and archived.                                                                                                                                                                                                                                                                                                                                                                           | 3.4.3     |       | ✓     | ✓   | ✓    |     |
| dd | Enquire if logs for successful and unsuccessful login attempts are regularly followed up. Examine and evaluate any evidence thereof.                                                                                                                                                                                                                                                                                                    |           | ✓     | ✓     | ✓   | ✓    |     |
| ee | Confirm that all default vendor accounts shipped with the operating system have been disabled.                                                                                                                                                                                                                                                                                                                                      |           | ✓     | ✓     | ✓   | ✓    |     |
| ff | Confirm that all dormant accounts have been disabled.                                                                                                                                                                                                                                                                                                                                                                               |           | ✓     | ✓     | ✓   | ✓    |     |
| gg | Confirm that the accounts of users that are on extended vacations have been disabled.                                                                                                                                                                                                                                                                                                                                                  |           | ✓     | ✓     | ✓   | ✓    |     |
| hh | Confirm that accounts are not set-up to run single commands. (If such accounts exist ensure that they make use of the restricted shell rsh.)                                                                                                                                                                                                                                                                                                 |           | ✓     | ✓     | ✓   | ✓    |     |
| ii | Confirm the propriety of all files and directories used in init processing                                                                                                                                                                                                                                                                                                                                                                   |           | ✓     | ✓     | ✓   | ✓    |     |
| jj | Enquire whether root users are aware of and diligent against vt escape (“mail bomb”) type attacks.                                                                                                                                                                                                                                                                                                                                  |           | ✓     | ✓     | ✓   | ✓    |     |

68
<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>CHAP Ref.</th>
<th>Compl</th>
<th>Valid</th>
<th>Acc</th>
<th>Cont</th>
<th>Int</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>File System Security</td>
<td>3.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>General</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>Confirm that there are no <code>.exrc</code> files on the system that are not necessary.</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>Confirm that all <code>.exrc</code> files that do exist have permissions set to read and write only by the owner.</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>Confirm that <code>.forward</code> files in user home directories can not execute a command or run a program.</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>Confirm that the following files’ permissions are set to what is denoted next to it:</td>
<td>3.4.2</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>/etc/utmp</code> 644</td>
<td>3.4.2</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>/etc/sm</code> 2755</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>/etc/sm.bak</code> 2755</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>/etc/state</code> 644</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>/etc/motb</code> 644</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>/etc/mtab</code> 644</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>/etc/syslog.pid</code> 644</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>Confirm that the <code>setgid</code> privileges have been removed from <code>/usr/kvmcrash</code>.</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>Confirm that the kernel is owned by root, its group set to “0” and permissions are set to 644.</td>
<td>3.4.2</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g</td>
<td>Evaluate the <code>umask</code> values for users and determine the appropriateness thereof.</td>
<td>3.4.2.1</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h</td>
<td>Determine what files have the <code>suid</code> and <code>sgid</code> enabled. Evaluate the necessity of each.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>i</td>
<td>Enquire whether <code>suid</code> and <code>sgid</code> shell scripts are allowed on the system?</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>j</td>
<td>Identify and list all programs <code>suid</code> to root.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>k</td>
<td>Identify, document and evaluate the security controls over all “non-standard” <code>suid</code> and <code>sgid</code> programs.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>l</td>
<td>Confirm that all <code>suid</code> programs to root have file permissions set so that it is only writeable by the owner.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>m</td>
<td>Review all <code>suid</code> and <code>sgid</code> programs for proprietary.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Identify all programs <code>suid</code> to other system logins.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o</td>
<td>Confirm that all <code>suid</code> login programs have file permissions set so that it is only writeable by the owner.</td>
<td>3.4.2</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>p</td>
<td>Confirm that directory modes in the path to all <code>suid</code> login programs prevent write access.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>q</td>
<td>Identify all programs <code>sgid</code> to system groups.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>Confirm that all <code>sgid</code> programs to system groups have file permissions set so that it is only writeable by the owner of the group.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s</td>
<td>Confirm that directory modes in the path to all <code>sgid</code> group programs prevent write access.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Description</td>
<td>CHAP Ref.</td>
<td>Compl</td>
<td>Valid</td>
<td>Acc</td>
<td>Cont</td>
<td>Int</td>
</tr>
<tr>
<td>----</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>t</td>
<td>Confirm that no <code>sgids</code> to system group programs are shell scripts.</td>
<td>3.4.2.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>u</td>
<td>Review <code>sgid</code> to system groups for propriety.</td>
<td>3.4.2.3</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>Determine if the creator's <code>gid</code> or <code>dir gid</code> is used for new files.</td>
<td>3.3.3.1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>w</td>
<td>Identify, document and evaluate procedures that are in place for checking system mounts for <code>suid/sgid</code> programs.</td>
<td>3.4.2.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>x</td>
<td>Confirm that the <code>chown</code> and <code>chgrp</code> commands reset the <code>suid/sgid</code> bit.</td>
<td>3.4.2.3</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>y</td>
<td>Confirm that the <code>suid</code> bit has been removed from <code>/usr/etc/restore</code></td>
<td>3.4.2</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>Confirm that the <code>.cshrc</code> files have access rights set to read and write only by the owner of the file.</td>
<td>3.4.2</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aa</td>
<td>Confirm that the <code>.login</code> files have access rights set to read and write only by the owner of the file.</td>
<td>3.4.2</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bb</td>
<td>Confirm that the <code>.logout</code> files have access rights set to read and write only by the owner of the file.</td>
<td>3.4.2</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cc</td>
<td>Evaluate the modes in use for the <code>cron</code> directories and files.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>dd</td>
<td>Confirm the propriety of users allowed to use <code>cron</code> services.</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ee</td>
<td>Confirm the propriety of <code>cron</code> fields for root and system logins.</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ff</td>
<td>Confirm that no read access rights are granted on <code>/dev/tty*</code> to group or other.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>gg</td>
<td>Confirm that <code>/dev/kmem</code> and <code>/dev/swap</code> are not accessible by other.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>hh</td>
<td>Confirm that <code>/dev/kmem</code> is owned by root (<code>uid=root</code>) and if group <code>kmem</code> exists <code>grp=kmem</code>.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ii</td>
<td>Confirm that if group <code>kmem</code> exists <code>ps</code> type utilities have <code>sgid=kmem</code> and mode set to 640.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>jj</td>
<td>Confirm that if no group <code>kmem</code> exists <code>ps</code> utilities have mode set to 600.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>kk</td>
<td>Confirm that <code>/dev/drums</code> not readable by other.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ll</td>
<td>Confirm that all device files other than the current <code>ttys</code> are owned by root and a system group.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

### 4. PROGRAM CHANGE CONTROLS

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>CHAP Ref.</th>
<th>Compl</th>
<th>Valid</th>
<th>Acc</th>
<th>Cont</th>
<th>Int</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Confirm that the development and production machine do not regard each other as trusted hosts.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>b</td>
<td>Confirm that developers do not have root access rights on any system especially in the testing and production environment.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>c</td>
<td>If development takes place on a separate machine identify if this system is linked to the other system in any way. (e.g. leased lines)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>d</td>
<td>If development takes place on the production machine identify, document and evaluate how the development and production environments are separated.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
4.5 CONCLUSION

Generally all the components identified have facilities that are aimed at computer security. Most of the components identified have as their primary objective the performance of specific tasks for which they have been designed. The security features that are provided are of secondary importance.

Because of the above mentioned and the way in which UNIX is developed the security features imbedded in the different components are not always easy to identify and review and generally not easy to maintain.

In UNIX most of the system tasks are performed by different so called programs. This has the effect that there are little, or as in most cases no interaction on a security level between components. This makes it difficult to decide which components should be installed or used. What makes the tasks of implementing security in UNIX even more daunting is the fact that all the security features in a specific component cannot be implemented at once as this is both inefficient and impractical.

Doing an installation review on a UNIX system can therefore also be a difficult and cumbersome task. Because many UNIX suppliers ship their own system tools and programs with UNIX, and UNIX security weaknesses are well known, documented and appear very often, the task of evaluating a UNIX systems becomes even more difficult.

It is therefore up to management to implement and enforce its security philosophy. This can most effectively be done by using security packages to compliment standard UNIX security features. CERT advisories should regularly be checked for known problems of the UNIX version in use at an organisation and patches that have been certified must be installed to rectify security weaknesses.
# CHAPTER 5

## CONCLUSION

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>73</td>
</tr>
<tr>
<td>2. OVERALL ORGANISATIONAL POLICY</td>
<td>73</td>
</tr>
<tr>
<td>3. APPLICATION AND USE</td>
<td>73</td>
</tr>
<tr>
<td>4. EVALUATION</td>
<td>74</td>
</tr>
<tr>
<td>5. CONCLUSION</td>
<td>75</td>
</tr>
</tbody>
</table>
CONCLUSION

1. INTRODUCTION

The main aim of this chapter is to evaluate whether the Access Path Model can be successfully applied to the predefined UNIX environment and if the audit programme developed can be used to evaluate and analyse control procedures in this environment.

2. OVERALL ORGANISATIONAL POLICY

The control model and framework for evaluating UNIX security must not be seen in isolation but rather as part of the overall organisational security policy. It is the responsibility of management to develop, implement and maintain this security policy. Management fulfill this duty of theirs by identifying the security risks faced by the organisation, evaluating the possible impact thereof on the organisation, forming and evaluating a solution for the security threats, setting out the principles, objectives and requirements of information processing, and formalising the above mentioned by developing a security policy.

The security policy has then to be put in operation, applied and monitored. This is done by a security team appointed by management.

3. APPLICATION AND USE

The persons that are responsible for the operation of the security policy set by management should have a clear view of the environment they are dealing with.

The UNIX operating system consists of numerous components (programs), some of which may even perform the same system function but controls supplied by the programs differ, this makes the application of Access Path and Path Context models advantageous and effective for the evaluation of controls in the system.

Each layer in the access path provides opportunities for the evaluation of controls pertaining to the different components in that layer. In this essay only those components in each layer that were considered to be a major component was evaluated. Further detail of the different layers especially the network layer could form the bases for further research.
4. EVALUATION

Using the methodology established in this research essay, and the Access Path and Path Context Models, it was possible to evaluate the predefined UNIX environment and develop an audit programme for the review of a UNIX installation.

It was found that in each of the different layers, except for the networking layer of the predefined path there was sufficient controls to ensure that:

- access overall are restricted to system objects on a per user or per group basis, as well as controls to limit the propagation of access rights;
- users are only allowed on networks after they have uniquely identified themselves; and
- an audit trail of all access of system objects is maintained.

The controls set out above complies with class C2 security as set out in the Orange Book. Additional controls were also identified to meet the other control objectives previously set.

The networking layer and the components of this layer that have been discussed in this work does not comply with the following control objectives as set out in table 5.1

<table>
<thead>
<tr>
<th>NO</th>
<th>DESCRIPTION</th>
<th>CHAPTER REFERENCE</th>
<th>COMPL</th>
<th>VALID</th>
<th>ACC</th>
<th>INT</th>
<th>CONT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>UDP</td>
<td>3.2.3</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>TCP</td>
<td>3.2.3</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>TELNET</td>
<td>3.2.5</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td>RLOGIN</td>
<td>3.2.6</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>e.</td>
<td>RSH</td>
<td>3.2.6</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>f.</td>
<td>UUCP</td>
<td>3.2.8</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g.</td>
<td>FTP</td>
<td>3.2.9</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>h.</td>
<td>ANON. FTP</td>
<td>3.2.9.1</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>i.</td>
<td>FINGER</td>
<td>3.2.10</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>j.</td>
<td>REXEC</td>
<td>3.2.11</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k.</td>
<td>TRUSTED</td>
<td>3.2.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

TABLE 5.1: Control objectives not met in the Networking layer of the model developed
Because of the impact of the security weaknesses of the network layer on the rest of controls in the system, it was found that the operating system has more risk factors than controls to control these risks. It is therefore concluded that all control objectives will not be met by the UNIX operating system if any of the components in table 5.1 is active on the system without the use of an external computer security package to compliment the controls of the UNIX operating system.

Because each layer in the model consists of different components, all of which do not need to be active for the UNIX operating system to function properly, the auditor/system administrator must consider whether the components active on the installation under review, are required and if so, the effect thereof on the overall computer security, as some of these components present the installation with risk factors that cannot be effectively and efficiently countered by any of the layers in the model.

Generally all the system components evaluated provided facilities that enable the system administrator to implement some security control features. However, the system software components have as their main purpose, the performance of specific tasks for the system, while maintaining maximum availability. The security features that these components do provide are of secondary importance.

Accordingly, are these components' security features not always designed in a fashion that will facilitate their review, and are generally not easy to maintain. In addition, the implementation and maintenance of security features that are supplied with the components always require a positive action from the security personnel, that is, free access at all levels, unless there is a restriction placed on specific users or resources.

5. CONCLUSION

In the light of the above, the objectives of this research essay have been met.

As a result of the successful application of the Access Path and Path Context Models to the specific environment, a positive contribution to the field computer security has been made.

The applicability of the Access Path and Path Context models to the predefined UNIX environment has been proved. The universal applicability of these models to other computer systems has not yet been proved and can form the basis for future research.
BIBLIOGRAPHY

8LGM 1991: Advisory 1 UNIX. Internet @ftp.army.com

8LGM 1991: Advisory 2 UNIX. Internet @ftp.army.com

8LGM 1991: Advisory 3 UNIX. Internet @ftp.army.com

8LGM 1992: Advisory 4 UNIX. Internet @ftp.army.com

8LGM 1992: Advisory 5 UNIX. Internet @ftp.army.com

8LGM 1994: Advisory 6 UNIX. Internet @ftp.army.com

8LGM 1994: Advisory 7 UNIX. Internet @ftp.army.com

8LGM 1993: Advisory 9 UNIX. Internet @ftp.army.com

8LGM 1992: Advisory 11 UNIX. Internet @ftp.army.com

8LGM 1991: Advisory 12 UNIX. Internet @ftp.army.com

8LGM 1994: Advisory 15 UNIX. Internet @ftp.army.com

8LGM 1995: Advisory 17 UNIX. Internet @ftp.army.com

8LGM 1995: Advisory 22 UNIX. Internet @ftp.army.com

8LGM 1995: Advisory 24 UNIX. Internet @ftp.army.com

8LGM 1995: Advisory 17 UNIX. Internet @ftp.army.com

ARMSTRONG, J C; DOUBA, S M; LEE HENRY, S; ROSE, R; RUMMEL, R E; PARKER, S; MARSHALL, A; DIPPOLD, R; NEGUS, C; VALLEY, J; SMITH, J; TAYLOR, D; WEINSTEIN, S S; TILL, D; PEPPARD, S & HOLSBERG, P 1994: UNIX unleashed; first edition. USA: SAMS Publishing.


BRITISH STANDARDS INSTITUTION 1993: A code for practice for information security management; London: BSI.

CURRY, D A 1992: UNIX system security, a guide for users and system administrators; Addison Wesley Publishing Co.

CURRY, D A 1990: Improving the security of your UNIX system; Internet @UK.OMS.COM


ERNEST & YOUNG INTERNATIONAL LTD. 1994: Audit, control and
security features of the UNIX operating system.

FARMER, D 1995: improving the security of your site by breaking into it. Internet zen@sun.com

FARROW, R 1991: UNIX system security, how to protect your data and prevent intruders. Addison Wesley Publishing Company


GROFF, JAMES, WEINBERG & PAUL 1983: Understanding UNIX a conceptual guide. QUE Corporation


MUFFET, A 1994: Computer security frequently asked questions. Internet, @UK.Sun.COM

REINHARDT, R B 1993: an architectural overview of UNIX network security. Internet, breinhar@access.digex.com.

SAICA. 1989: Auditing in a computer environment, audit and accounting guide. SAICA.

US DEPARTMENT OF ENERGY 1990: CIAC A-20 UNIX Internet attack advisory. Internet webmaster@ciac.llnl.gov.


US DEPARTMENT OF ENERGY 1994: CIAC F-09 UNIX /bin/login vulnerability. Internet webmaster@ciac.llnl.gov.