國立交通大學

資訊學院 資訊學程

碩士論文

植基於虛擬化技術之網路安全縱深防禦架構

Network Security Defense-in-Depth Architecture
based on Virturalization Technology

研究生：李長霖

指導教授：蔡文能　教授

中華民國一百年六月

植基於虛擬化技術之網路安全縱深防禦架構
Network Security Defense-in-Depth Architecture based on Virtualization Technology

研 究 生：李長霖　　Student：Chang-Lin Lee
指導教授：蔡文能　　Advisor：Wen-Nung Tsai

國 立 交 通 大 學
資訊學院 資訊學程
碩 士 論 文

A Thesis
Submitted to College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science
June 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年六月

植基於虛擬化技術之網路安全縱深防禦架構

學生：李長霖　　指導教授：蔡文能教授

國立交通大學　資訊學院　資訊學程碩士班

摘　要

IBM 公司在 1960 年發展主機虛擬化技術以來，虛擬化技術已經變成愈來愈流行。之後不少的學術或是商業研究紛紛提出不少虛擬化技術之產品及應用，像是 VMWare 公司所推出的 ESX Server 以及 Microsoft 公司所推出的 Hyper-V 技術等，當愈來愈多虛擬化技術的應用被提出後，我們開始思考如何將其運用在網路安全防護架構部署上。

本篇論文所提出的一個植基於虛擬化技術之網路安全縱深防禦架構解決方案，可以有效的降低企業在部署縱深防禦架構上之成本並且提高電腦資源之使用率。另外本篇論文也將針對傳統縱深防禦架構、整合威脅管理系統及本研究所提之虛擬化縱深防禦架構做一綜合性的深入研究，並比較其優缺點，另外也針對上述架構進行網路效能測試之分析探討。冀望本篇論文能對虛擬化網路安全相關研究以及企業內部的縱深防禦建置提供貢獻。

關鍵字：縱深防禦、虛擬化、效能測試、網路安全。

Network Security Defense-in-Depth Architecture
based on Virtualization Technology

Student：Chang-Lin Lee Advisor：Prof. Wen-Nung Tsai

Degree Program of Computer Science
National Chiao Tung University

ABSTRACT

In 1960, IBM mainframe Virtualization Technology has been developed, it has become more popular. Many academic or commercial research have developed many virtualization products and applications, such as VMWare corporation has released VMWare ESX Server and Microsoft Corporation has released a Hyper-V technology.When more Virtualization Technology is released, we started thinking about how to use the network security architecture deployment.

In this paper, we propose a solution “Network Security Defense-in-Depth Architecture based on Virtualization Technology” that can effectively reduce cost of the deployment of defense in depth and increase the usage of computer resources. Further, this paper will also compare their advantages and disadvantages for the traditional defense in depth architecture, Unified threat management and our solution. While also analysis their network performance. We hope this paper can provide a contribution in virtualization network security research and the defense in depth research.

Keywords:Defense-in-Depth,Virtualization,Performance Benchmark, Network Security.

誌　　謝

經過了交通大學資訊學院資訊組碩士班二年的努力，克服了種種困難，終於完成了我的碩士論文。這篇論文之所以能順利的完成，首先要感謝我的指導教授 — 蔡文能教授。由於他不辭辛勞地在我論文寫作及口試期間，給予我知識上的啟發以及論文撰寫上的指導，我才能夠順利的拿到碩士學位。還有學長及宣佑、嬿竹、彥皓及學弟兆民的熱心討論與幫忙，這篇論文才能順利的完成，還要感謝資策會的長官同事的幫忙，很多硬體資源都是和公司借來實測的，如果沒有這些資源在手，相信我的實驗是無法做到完善的，最後感謝我的家人及未婚妻辰妘在背後的默默支持，使我無後顧之憂能夠順利完成學業。

目　　錄

表　目　錄

圖　目　錄

第一章　緒論(Introduction)

近幾年來，隨著軟硬體技術快速進步，且電腦網路之應用也愈趨成熟，使得人們可隨時隨地利用電腦進行網路購物、即時查詢電子地圖、電子信件及視訊交談等。但隨之而來的問題即是安全性問題，駭客能夠運用各式各樣的攻擊手法竊取個人資料或是進行遠端遙控，傳統的網路閘道式防火牆已無法有效阻擋這類型的攻擊，因此有各式各樣的防護手法如雨後春筍般一一出現，像是入侵偵測系統、防毒牆等，目的皆是為了降低企業用戶洩漏機密資料的風險，但是皆有其防護限制，因此為了達到符合資訊安全三項基本特性：機密性(Confidentiality)、完整性(Integrety)及可用性(Availability)，縱深防禦的觀念開始被運用在網路安全的防護上，其特性在於利用一層一層的防護過濾技術，有效的將防禦線拉大，每個防護設備各司其職進行阻擋過濾的工作，因此更能有效的阻擋攻擊事件的發生。

由於軟硬體爆炸性的成長，虛擬化技術近年來被不斷拿出來討論，虛擬化技術能夠使資源更有效的被運用，通常網路安全防護設備常常資源僅用到其最大資源的 20~30%，且升級擴充也不容易，如果遇到突如其來的大量封包癱瘓攻擊，往往因單一失效點(Single Point of Failure)的問題造成企業嚴重損失，因此本論文將提出一種作法，利用虛擬化技術建構縱深防禦架構，其優點有三，一是可有效運用資源，二是能節省成本，包含機架空間、電力成本等，三是可符合網路安全基本三項特性。

1.1. 研究動機與目的

在一個大型企業當中，通常部署著非常多的網路安全防護設備，像是防火牆、入侵偵測系統以及防毒牆等，而本研究動機主要是因為在管理這樣的網路常會遇到一個問題，就是機架空間不足，且電力成本相當可觀，在現今注重綠化的世界，如果能將這些防護設備運行於同一硬體平台架構，資源可以共享，就可以節省實體機架空間的不足、降低電力成本及節省能源使用，且也能夠達到與實體防護同樣等級的安全防護效果，因此本研究的目的旨在「比較實體縱深防護架構與虛擬化的縱深防禦架構之網路效能分析」。如果分析的結果不會相差太大，而外顯的成本及能源使用量卻能因此降低，相信對於企業來說是非常具有吸引力的。

1.2. 研究範圍

本研究將以 Vmware ESX Server 4.1 為此一虛擬化網路安全縱深防禦架構之系統平台，並在其上搭配 Linux 作業系統以及其他開放原始碼軟體(OpenSource)架設防火牆、入侵防禦系統及防毒牆來建構虛擬化的縱深防禦網路安全防護架構，最後利用 Httperf 及 Iperf 網路效能測試工具對此虛擬化網路安全縱深防禦架構、傳統網路安全縱深防禦架構以及整合威脅管理系統進行效能分析之結果比較。

1.3. 研究架構

本論文共分六個章節，架構分別說明如下：

第一章 緒論

說明研究背景、動機、目的、研究範圍及方法。

第二章 背景知識

本章將介紹目前常見的網路安全攻擊手法、網路安全防護手法、縱深防禦機制的概念、虛擬化技術之種類說明以及網路效能測試的方法。

第三章 相關研究

本章將介紹一些目前在網路安全防護應用、縱深防禦機制應用以及虛擬化技術應用中較具代表性的研究成果。

第四章 植基於虛擬化技術之網路安全縱深防禦架構

本章將介紹運用虛擬化技術之網路安全縱深防禦架構，說明其系統概述、系統運作流程、系統內部組成模組及建置說明。

第五章 網路安全防禦機制之效能測試分析

本章針對第四章的內容與實體縱深防禦機制及整合式威脅管理設備進行效能測試分析。

第六章 結論

對本論文所達成的功能及效用做一結論，並探討未來可繼續進行的研究方向，供未來學者參考。

第二章　背景知識(Background)

當前駭客攻擊手法伴隨著科技發展日新月異，加上政府部門、企業及個人不時傳出遭到駭客侵擾或竊取個人資料等資安事件，嚴重影響組織的網路安全，因此本章將於 2.1 節介紹常見的網路安全攻擊手法，並於 2.2 節介紹有那些網路安全防護機制在保護我們的組織，接著在 2.3 節將對縱深防禦這個專有名詞進行說明，最後於 2.4 節介紹目前最熱門的虛擬化技術種類及說明。

2.1.　常見網路安全攻擊手法

所謂”知己知彼，百戰百勝”。若要了解如何防禦網路上駭客的攻擊，就必須要先了解駭客的攻擊是透過什麼技術、弱點來達成他們的目的。藉由瞭解駭客的入侵攻擊手法，有助於提昇企業在網路安全的應變能力，保障系統資源安全，底下將說明目前常見的網路安全攻擊手法。

1、IP偽冒攻擊(IP Spoofing)：

IP 偽冒攻擊簡單來說就是偽造網路封包的 Header，讓路由器（Router）或者是防火牆（Firewall），以為是來自安全可信任的網域，而被允許進入內部網路，直接攻擊網路主機。

2、連接埠掃描（Port Scan）

網路通訊埠掃描常是駭客攻擊前的資料蒐集手段。主機在提供連結網路服務的時候，像是網頁服務（Port 80），都要經過連接埠的連接，才能使用主機所提供的服務。駭客通常都會掃描目標主機的網路通訊埠(Port)來檢視主機提供那些服務，如果提供服務的程式有漏洞未經修補，駭客就能利用該服務的漏洞進行攻擊。

3、阻絕服務攻擊（Denial of Service Attack，DoS Attack）：

阻絕服務攻擊是藉由大量的封包或 TCP/IP 協定的漏洞，消耗網路的頻寬或耗竭系統資源，例如：CPU、Memory 等，阻撓系統或網路運作。常見的 DoS 攻擊分為：

(1).網路層攻擊：

駭客利用 TCP/IP 網路層（Network Layer）的封包，例如：IP、ICMP、IGMP 等封包攻擊目標主機，包括：Smurf Attack、Ping of Death Attack 等。

(2).傳輸層攻擊：

駭客利用 TCP/IP 傳輸層（Transport Layer）的封包，例如：TCP、UDP 等封包攻擊目標主機，包括：TCP SYN Flood Attack、UDP Flood Attack(Fraggle Attack)。

(3).應用層攻擊：

駭客利用 TCP/IP 應用層(Application Layer)的封包，例如：HTTP Request 封包攻擊目標主機，例如像 HTTP Request Flooding Attack。

4、分散式阻絕服務攻擊（Distributed DoS Attack，DDoS Attack）：

分散式阻絕服務攻擊指駭客能夠從遠端遙控僵屍網路(Botnet)中的僵屍電腦(Zombie)，並同時對所有僵屍電腦下達攻擊指令，每台僵屍電腦僅提供少量的攻擊封包，因此不易察覺自身電腦已被控制，且只要僵屍網路的僵屍電腦數量夠龐大，即可利用超量的攻擊封包癱瘓目標主機，導致系統無法提供正常服務。

5、惡意程式攻擊(Malware Attack)：

早期有非常多的攻擊手法，如電腦病毒、蠕蟲攻擊、木馬程式

等，依其攻擊目的而有不同的名稱，現今因不能清楚劃分其攻擊手法，因而將所有惡意的攻擊程式統稱為惡意程式(Malware)，底下將介紹目前主流之惡意程式。

(1).電腦病毒（Virus）

電腦病毒是指故意設計來干擾電腦的作業、紀錄、毀損或刪除資料，或散佈至其他電腦與網際網路上流竄的軟體程式，這類的軟體程式通常會減緩電腦的運作速度，並在過程中造成其他問題。電腦病毒會將本身複製到其他乾淨的檔案或是開機磁區，當電腦使用者沒注意的情況下，執行到已受到病毒感染的檔案或程式，病毒就會以相同的方式散佈或共用出去。

(2).電腦蠕蟲（Worm）

電腦蠕蟲和電腦病毒相似，是一種能夠自我複製的電腦程序。蠕蟲程序常駐於一台或多台電腦中，本身會複製許多分身，像蠕蟲般在電腦網路中感染多台電腦。電腦蠕蟲通常會掃描其他機器是否感染同樣的蠕蟲病毒，如果沒有被感染，就會透過內建的傳播手段進行感染，以達到使電腦癱瘓的目的。

(3).木馬程式（Trojan Horse）/後門程式(Backdoor)

木馬程式是一種由遠端操控被攻擊主機的軟體。當被攻擊主機被安裝後門程式之後，駭客就可以從遠端控制這台電腦，例如上傳/下載檔案、竊取密碼、更改視窗系統中的 Registry 或是將螢幕畫面傳回等。後門程式甚至可以與正常的執行檔結合，使不慎執行該程式的使用者在不知不覺

中被植入後門程式。

6、緩衝區溢位（Buffer Overflow）

緩衝區溢位是由於程式撰寫的疏忽，使得駭客利用作業系統或應用程式之程式設計上的缺失而進行攻擊，造成緩衝區容量的不足，導致系統執行駭客欲執行的攻擊程式。例如：處理一個長度為 80 字元的字串，程式設計師設定字串變數長度為 100，如果駭客將一個長度超過 100 的字串送入程式中處理，這樣超出的變數就可能會覆蓋其他程式的片段，造成程式執行無效。

7、重送攻擊(Replay Attack)

由於區域網路是採用廣播式(Broadcast)的網路，駭客可利用此特點竊聽同一個網路裡面的封包，即使是加密的通訊協定，若認證過程設計不良，或是所使用的加密演算法複雜度不夠高，也還是有可能會被破解。例如：早期當使用者以微軟的網路芳鄰通訊協定傳送檔案時，駭客可先截取封包，以獲得封包上的認證資訊，並在接下來的十幾分鐘內繼續使用這個認證通過網路芳鄰服務提供者的認證程序。

8、中間人攻擊(Man-in-the-Middle Attack)

已經存在的網路連線也有可能被第三者中途劫走（Session Hijacking）。例如 TCP 通訊協定的 3-way handshaking 就存有 Sequence Number 容易被猜測的弱點，或是像是 ARP Spoofing Attack 及 SSL Strip Attack 也是中間人攻擊的手法。

9、Web 網頁技術相關攻擊

網頁設計上的漏洞也會造成許多安全問題。許多動態網頁的技

術如 ASP、PHP 及 Javascript 等可以將網站由靜態網頁變成動態的、且可即時產生回應的模式。當網站程式設計不良時，很可能就造成駭客的可乘之機。駭客可以利用像是 SQL Injection、Cross-Site Scripting 等網頁攻擊技術從遠端直接閱讀網站主機上的任何資料或是執行任何遠端程式。

2.2. 網路安全防護機制

在網路安全防護機制中網路閘道端存取控管機制是一個重要的環節。內部網路和外部網路之間需建立起一道屏障，以防護網路邊界上的存取行為，將駭客阻絕於門外。網路所有存取行為都應被檢查或授權，才能夠進入企業網路。底下將介紹一般企業在部署網路安全防護機制時所使用到的防禦措施。

2.2.1. 防火牆系統(Firewall System)

防火牆[18]其主要定義為用以阻擋非法的連線，允許合法的連線進入企業內部的安全防護系統，其最基本的功能就是控制在電腦網路中，不同信任程度區域間傳送的資料流。例如網際網路是不可信任的區域，而內部網路是高度信任的區域。用以避免安全策略中所禁止的通訊服務。它能控制在不同信任區域之間的資料流。常見的區域包括網際網路(不信任的區域) 、內部網路(一個高度信任的區域)及非軍事管制區(Demilitarized Zone，DMZ)，如圖 1 所示。其最終目的是透過安全政策的執行提供不同信任區域之連通性以符合最小權限原則(Least Privilege)。

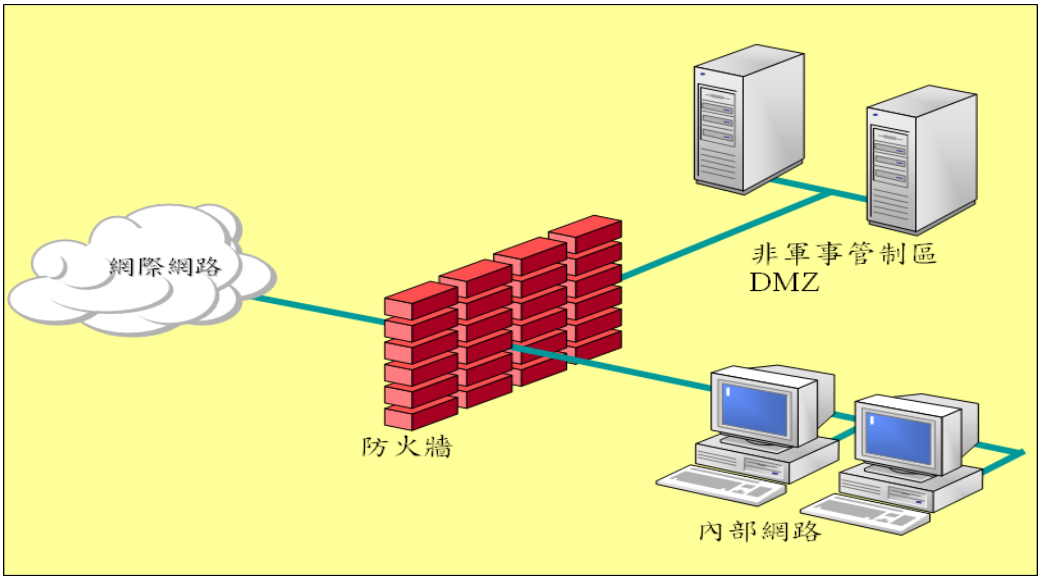

圖 1 企業防火牆架構示意圖

防火牆的類型：

1、個人防火牆

通常是在一部電腦上具有封包過濾功能的軟體，如 ZoneAlarm 及 Windows XP SP2 後內建的防火牆程式即稱之，可阻擋非法的連線直接連接到本機，屬於用戶端的防護措施。

2、網路層防火牆

網路層防火牆可視為一種 IP 封包過濾器，運作在底層的 TCP/IP 協定堆疊上。我們可以以列舉的方式，只允許符合特定規則的封包通過，其餘的一概禁止穿越防火牆。這些規則通常可以經由管理員定義或修改，而此種類型的防火牆通常都會做成硬體式的網路設備或是也能利用擁有 2 張以上網路卡的電腦自行建置，常見的網路層防火牆軟體為 Linux 中的 Iptables 或是 FreeBSD 中的 IPFilter。較新式的防火牆能利用封包的多樣屬性來進行過濾，例如：來源 IP 位址、來源網路通訊埠、目的 IP 位址或網路通訊埠、服務類型(如 WWW 或是 FTP)。也能經由通訊協定、TTL 值、來

源的網域名稱或網段等屬性進行過濾。

3、應用層防火牆

應用層防火牆是在 TCP/IP 堆疊的「應用層」上運作，使用瀏覽器時所產生的資料流或是使用 FTP 時的資料流都是屬於這一層。應用層防火牆可以攔截進出某應用程式的所有封包，並且封鎖其他的封包(通常是直接將封包丟棄)。理論上，這一類的防火牆可以完全阻絕外部的資料流進到受保護的主機裡。此類防火牆藉由監測所有的封包並找出不符規則的內容，可以防範電腦蠕蟲或是木馬程式的快速蔓延。不過這個方法較不易實作，因軟體種類極其繁多，所以大部份的應用層防火牆通常都只會實作某種功能的過濾機制，例如：Linux 的 Iptables 防火牆只要外掛特定應用程式的模組就能夠變成應用層防火牆。

4、代理伺服器(Proxy)

代理伺服器（Proxy）也能像應用程式一樣回應輸入封包(例如連線要求)，同時封鎖其他的封包，達到類似於防火牆的效果。代理伺服器能夠增加駭客從外部網路竄改一個內部系統之困難度，且只要對於代理伺服器有良好的設定，即使內部系統出現漏洞也不一定會馬上造成安全上的問題。

防火牆經常搭配網路網址轉換(NAT) 的功能，其主要是將主機放置於防火牆之後，並且使用 RFC1918[17]所定義的「私人網址空間」，如表 1 所示，這樣可以避免駭客輕易取得企業整體網路拓撲架構，增加其攻擊的困難度。

表 1 RFC1918 定義之私人網址空間列表

Begin Address	End Address	Prefix
10.0.0.0	10.255.255.255	10.0.0.0/8
172.16.0.0	172.31.255.255	172.16.0.0/12
192.168.0.0	192.168.255.255	192.168.0.0/16

2.2.2. 入侵偵測/防禦系統(Intrusion Detection/Prevention System)

如 2.1 節所介紹之網路攻擊的手法越來越多，有的充分利用防火牆放行許可，有的則使防毒軟體失效，例如：有一些作業系統的弱點已被發現但尚無解決之道，駭客利用這種全新的弱點攻擊企業的主機，這就是所謂零時差攻擊(Zero Day Attack)。

防火牆可以根據 IP 位址或網路通訊埠過濾封包。但是它對於利用合法的連線進行非法的行為並沒有阻擋能力。因為防火牆並不會深入封包包內檢查內容(Payload)。

防衛機制最好應該是在危害形成之前初期就起作用。因此發展出入侵偵測系統(Intrusion Detection System，IDS)這樣的防護措施，但僅能被動的通知被入侵往往無法即時阻擋災害的發生，因此後來延伸出結合防火牆系統進行阻擋，變成所謂入侵防禦系統(Intrusion Prevention System，IPS)，入侵防禦系統能夠對合法連線但是屬於非法行為的封包進行過濾及阻擋，其深入檢查網路封包並尋找它所認識的攻擊特徵代碼，過濾並丟棄有害的網路封包，最後進行記錄以便事後分析。

入侵偵測/防禦系統[15]依部署方式的不同主要可以分成兩大類：

1、網路型入侵偵測系統（NETWORK-BASED Intrusion Detection System，NIDS）

網路型的入侵偵測系統如圖 2 所示，主要是擷取每一個經過的網路封包作為資料來源，通常將網路卡設定為雜亂模式（Promiscuous Mode），來偵測分析流經網路層(Network Layer)的封包資訊。如果所偵測的封包資料與系統內置的安全規則吻合，入侵偵測系統就會發出警報。

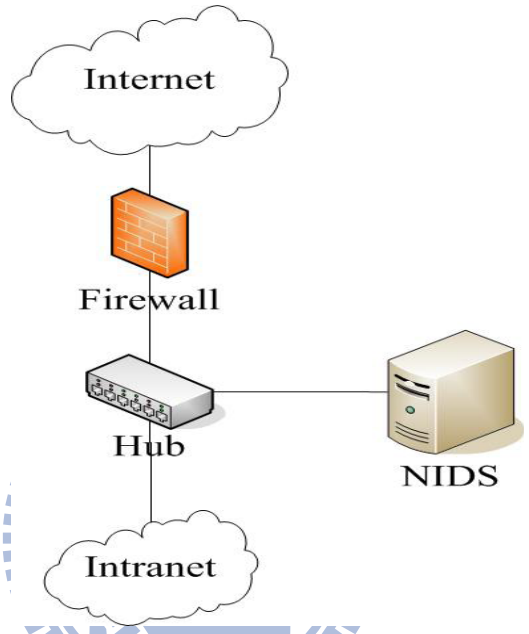

圖 2 網路型入侵偵測系統架構示意圖

2、主機型入侵偵測系統（HOST-BASED Intrusion Dection System，HIDS）

這種系統的軟體必須要直接載入主機並加以監控。主要偵測該主機的網路即時連接及系統檔案、程序與日誌檔中是否有可疑的活動。可以與網路型偵測結合使用，達成完整的入侵偵測。

入侵偵測系統依偵測方式的不同分為兩類：

1、誤用偵測系統（Misuse Detection System）

誤用偵測系統又稱為特徵型偵測（Signature-based Detection），檢測的方法類似電腦病毒的檢測方式，對已知的網

路攻擊手法及入侵行為，包含系統的缺陷，經由分析整理成入侵特徵（Signature）模式庫。入侵偵測系統藉由比對從主機或是網路中所蒐集到的資料特徵，是否在所收集到的入侵特徵模式庫中出現，如果有符合攻擊者入侵行為的特徵時，即發出警告（Alert）。透過入侵偵測模式庫，誤用入侵偵測系統可以詳細且準確的對已知的入侵行為發出警告訊息，因此有較低的誤判率(False Alert)，但對於未知的攻擊特徵則無法準確的偵測出來。

2、異常行為偵測系統(Anomaly Detection System)

異常行為入侵偵測系統主要在於先建立電腦或網路正常活動的統計記錄，將正常活動的統計記錄與現行的網路活動進行比較，如果現行網路活動有違反正常正常活動規律時，該活動即被認為是可能入侵行為。

此種偵測方式的優點在於可以檢測到未知或是較複雜的入侵行為，但誤報率(False Alert)也高。例如用戶行為的突然改變，極有可能被視為入侵行為。

2.2.3. 防毒牆系統(Viruswall System)

傳統的網際網路閘道安全解決方案通常是在區域網路出入網際網路的閘道端防火牆、入侵防禦系統的方式架構。而惡意程式攻擊這一部份主要依靠主機端安裝防毒軟體加以阻擋，但現今的惡意程式攻擊手法日新月異，常常利用一些巧妙的手法或系統漏洞就能夠從作業系統核心層將主機式防毒軟體關閉，造成企業的嚴重損失，因此就有了閘道式防毒牆的防護措施。

防毒牆主要是檢查封包所帶的應用程式內容之防護措施，而各式應用程式包羅萬象，因此目前常見的防毒牆僅針對較受歡迎的服務進行

檢測，例如可以針對下載的檔案在防毒牆重組後進行掃描的動作，如果沒有符合病毒碼即放行，因此利用防毒牆進行過濾，可減少使用者端電腦主機被感染的機率。市面上常見的防毒牆在七個主要的通訊協定中利用透通模式代理(Transparent Mode Proxy)線上掃描所有進來的封包，包括：網頁(HTTP)、郵件(SMTP、IMAP、POP3)、下載(FTP)，新聞(News)和立即訊息(Socks)等，可以提升企業內部網路安全防護等級。

2.2.4. 整合式威脅管理系統(Unified Threat Management)

整合式威脅管理系統是一個全面的解決方案，自 2004 年出現在網路安全行業以來，已成為企業首要的網路防禦系統。從理論上講，它是由傳統防火牆演變而來的一個多功能網路防禦設備，它能夠執行多種安全功能如：防火牆、入侵防禦、網路閘道端防毒、網路閘道端防垃圾郵件、內容過濾、VPN 及負載平衡等。

整合式威脅管理其優點在於管理多個防護系統，而不是只單獨處理像是防毒、內容過濾、入侵防禦和垃圾郵件過濾功能，企業可以靈活地部署單一 UTM 設備，所有的功能只要一個單機式網路設備即完成。而吸引中小企業之主要優點除了安裝佈署快速、管理方便、成本低廉外，還有就是封包只需要解開一次，傳統的縱深防禦架構每經過一層防禦系統就要重新解開一次封包，這是非常沒有效率的，但 UTM 的瓶頸在於其多合一的特性，該特性引發效能不佳的批評。甚至有企業在佈署 UTM 設備後發現網路效能嚴重下降，最後將 UTM 設備上特定的功能關掉，導致失去“多合一”的初衷。另外在發生阻斷服務攻擊時，系統處理大量的攻擊封包容易丟失部份封包，造成攻擊滲透的問題，UTM 設備強調整合性防禦多種安全機制於一點，卻反而因此缺少了縱

深防禦的特性。

2.3. 縱深防禦機制

構築縱深防禦的網路防護架構，是網路防護方案的核心原則。縱深防禦(Defense-in-Depth)原意是一種軍事戰略，有時也稱作彈性防禦或是深層防禦，是以全面深入的防禦去延遲前進中的敵人，透過放棄空間來換取時間與給予敵人額外的傷亡。當縱深防禦被運用在網路安全上時則意味著以多層網路安全技術減輕企業資訊安全風險。

舉例來說，網路防火牆、入侵防禦系統及個人電腦防毒軟體即形成了階層化(Layered Approach)的防護架構，層層阻擋網路惡意程式之攻擊。即便企業因資訊作業與與系統發展的需求，改變了網路防火牆原先規範的安全設定，進而影響網路邊界防護的有效性，縱深防禦架構中的其他機制仍可能發揮功效而達成防護效果。

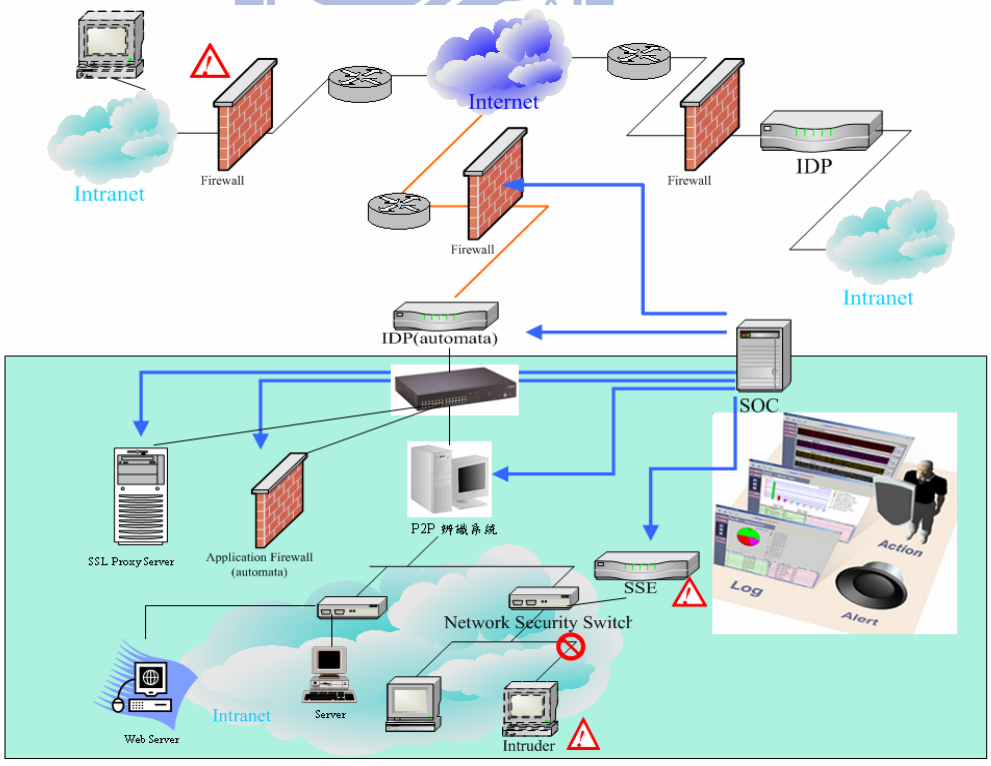

圖 3 網路安全縱深防禦系統架構

我們再以企業的邊界安全角度來考量，一般來說防火牆必須根據一套設定好的規則來過濾可疑的網路存取行為，但無法主動判斷是否有合法連線中的非法行為。因此為了彌補防火牆本身的限制，必須加入其他的網路安全防護機制進行防護，如圖 3 所示[13]，須在防火牆其前或後建置入侵防禦系統、內容過濾器及防毒牆等網路安全設備，以提升邊界網路的安全性。可降低因不當使用網路所造成的損害，避免因一種網路安全機制被駭客攻破，而使其能長驅直入，提高攻擊的困難度，進而提升企業網路安全的整體防護能力。

2.4. 虛擬化技術

在計算機科學中，虛擬化（Virtualization）是一個表現邏輯群組或電腦資源子集的行程(Process)，用戶可以用比原本的組態更好的方式來存取這些行程。這些資源的新虛擬部份是不受現有資源的架設方式、地域或物理組態所限制。一般所指的虛擬化資源包括計算能力和資料儲存。

虛擬化技術[19]（Virtualization Technology）可以讓一部電腦主機（host computer）建立與執行一至多個虛擬化環境（Virtual Environment），該技術多半使用實效模擬（Emulate）來模擬出一部完整的電腦系統（Computer System），之後再將作業系統（Operating System）安裝在這部虛擬化的電腦系統上，就作業系統角度而言此一虛擬化環境與真正實體電腦並無太大差異，可以按照傳統的操作方式來維護作業系統，而我們稱這樣的作業系統為子作業系統（Guest Operating System，Guest OS）。

虛擬化技術[5]有的是在既有作業系統上執行（如 QEMU、VMware Workstation、Virtual PC），有的則比作業系統更先安裝至電腦中（比

作業系統更具主體性，即在硬體與作業系統核心中再插入一 Hypervisor 層，如 Xen Server、VMware ESX Server 及 Microsoft Hyper-V Server），其主要是利用一部電腦主機模擬多個虛擬化環境，然而更先進者也能將多部電腦以虛擬化技術融合成單一的虛擬化環境，如 VMware vSphere Server，可同時管理多台 VMware ESX Server，且資源可以共享。

此外還有另一種並非以模擬出完整硬體以供子作業系統運作的虛擬化技術，其主要是模擬出一個相容於之前所發展出來的應用程式及驅動程式之虛擬化環境，此也屬虛擬化技術的範疇，如.NET 的 CLR、JAVA 的 JVM/JRE 皆屬於此類。例如用 Java 所寫的程式可以透過對 JAVA 執行環境(JRE)軟體發送命令以獲得服務，取得期望的結果。透過提供這種服務，JAVA 軟體起到了虛擬機器的作用。程式不必為特定的作業系統或硬體撰寫，提高程式可攜性。

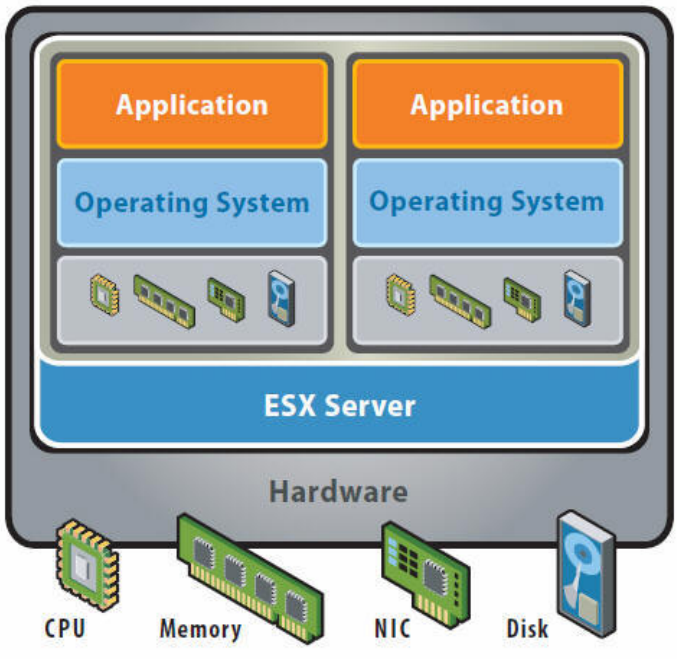

圖 4 虛擬化平台架構示意圖

虛擬機器根據它們的運用和與實體機器的相關性分為兩大類。系統虛擬機器提供一個可以執行完整作業系統的完整系統平台。相反，程式虛擬機器為執行單個電腦程式設計，這意味它支援單個行程。虛擬機器的一個本質特點是執行在虛擬機器上的軟體被局限在虛擬機器提供的資源裡——它不能超出虛擬世界。圖 4 為 VMWare 公司所推出之虛擬化技術的架構示意圖。

第三章　相關研究(Related Researches)

本研究主要是從各種不同領域之學術研究中所發展出來的一種全新網路安全防禦架構。因此將於底下各節介紹這些不同領域之相關學術研究，於 3.1 節介紹網路安全縱深防禦架構之相關研究，於 3.2 節介紹虛擬化技術之相關研究，最後於 3.3 節介紹網路效能分析之相關研究。

3.1.　網路安全縱深防禦架構之相關研究

在 2000 年時，中央大學曾宇瑞[14]提出一個網路安全縱深防禦架構，該篇論文最大的特點除了保留傳統防火牆機制外，尚引入入侵偵測系統(IDS)與陷阱機制，利用入侵偵測系統即時偵測入侵行為之功能，彌補陷阱系統被動呆版之缺點，亦利用陷阱系統隔離觀察之概念，彌補以往入侵偵測系統誤報率高之缺失，兩者取長補短並搭配防火牆形成一完美組合，使任何連線都要通過多層檢測，才能成功連上伺服器，其中，若連線之檢測被確認有入侵嫌疑，則嫌疑者會被導入陷阱系統(Deception system)，進行隔離觀察，而被導入觀察之連線，若經觀察一段時間發現沒問題，仍會被導回原正常服務，否則將予以斷線或作其他處置（例如 E-mail 警告來源主機）。整個縱深安全防護的概念如圖 5 所示，有四道防線，分述如下：

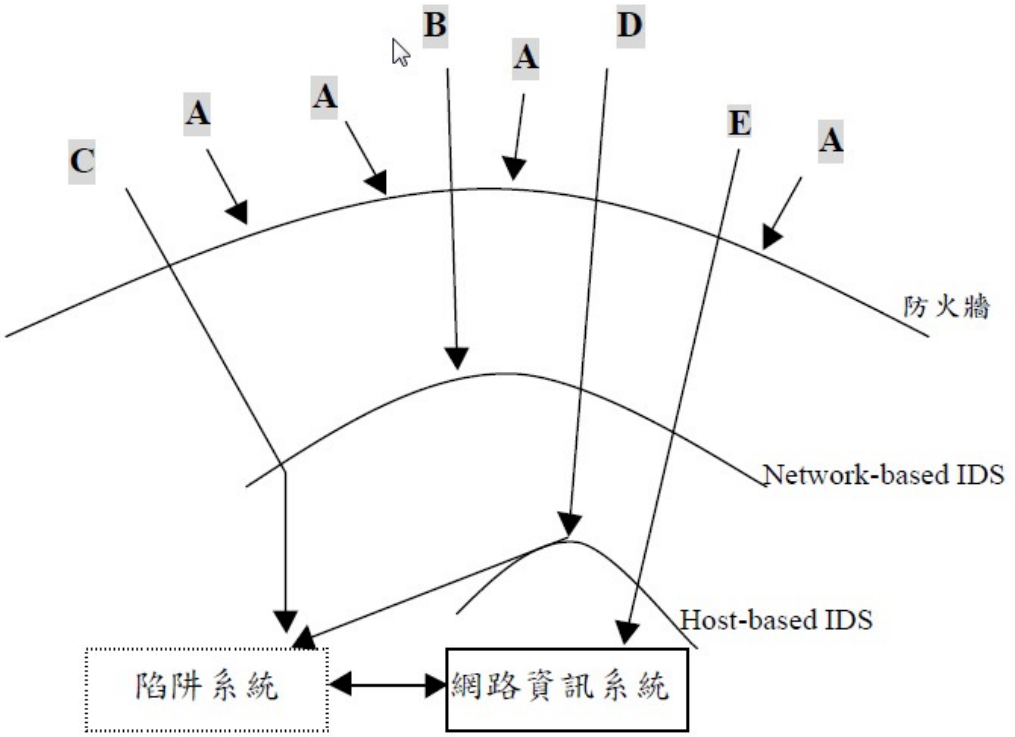

圖 5 縱深防禦架構示意圖

圖中各英文字的說明如下：

A 未經許可或未開放之通訊協定（服務）可直接由防火牆擋掉。

B 當 Network-Based IDS 確認攻擊行為發生後，除作成紀錄備查外，並指示防火牆將該連線中斷。

C 入侵嫌疑者被 Network-Based IDS 導入陷阱系統進行嚴密監視。

D 當 Host-Based IDS 以誤用偵測技術發現系統有不正常紀錄（例如：同一來源多次登入失敗），便將其導入陷阱系統進行嚴密監視。

E 正常之連線狀況。

在 2001 年 6 月時，Dorene Kewley 及 John Lowry 在 IEEE 期刊上提出一個很重要的概念[1]，他們認為網路安全縱深防禦(Defense in Depth)不能只著重在單一攻擊行為上，還要針對廣度防禦(Defense in

Breadth)進行部署考量，且在考量部署安全防禦機制時，要考慮佈署的層數不可太多，因為這樣反而會導致防禦產品太多造成維護過於複雜，而增加被駭客入侵的風險，如果只在網路邊界安全上做縱深防禦的規劃，對於企業內部環境防禦安全層級的提升是有限的，應該要在不同的地方部署相對應的防禦措施，如此才能確保企業在各面向上都做好安全防護，作者利用分組實驗，將攻擊者歸為紅隊，將防禦者歸為藍隊，並且針對機密性、完整性及可用性三個面向設定不同的情境進行四個階層的縱深防禦架構實驗測試，從實驗過程中可以得知，如果在防禦相同的事情上面，多增加一個防禦機制並不會提升整體防禦的效果，如圖 6 所示，在二層防禦架構下，駭客攻擊所花費的時間和成功進入系統的時間和三層防禦架構下有顯著的差異，但在三層防禦架構及四層防禦架構下，卻沒有差別太大。

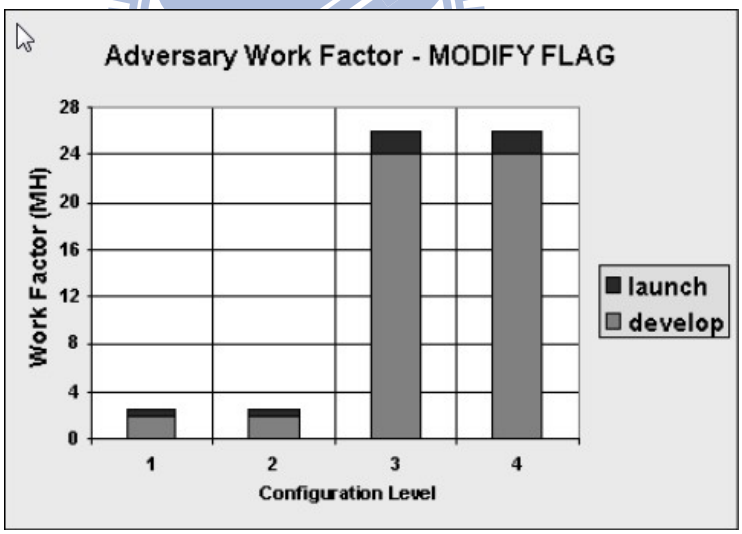

圖 6 縱深防禦層數及攻擊執行成功所花費之時間比較圖

但如果能夠對症下藥，根據各種不同的需求建構不同的防禦機制，如此才能有效避免被駭客入侵成功，作者最後的結論認為深度防禦和廣度防禦是同樣重要的，如圖 7 所示。

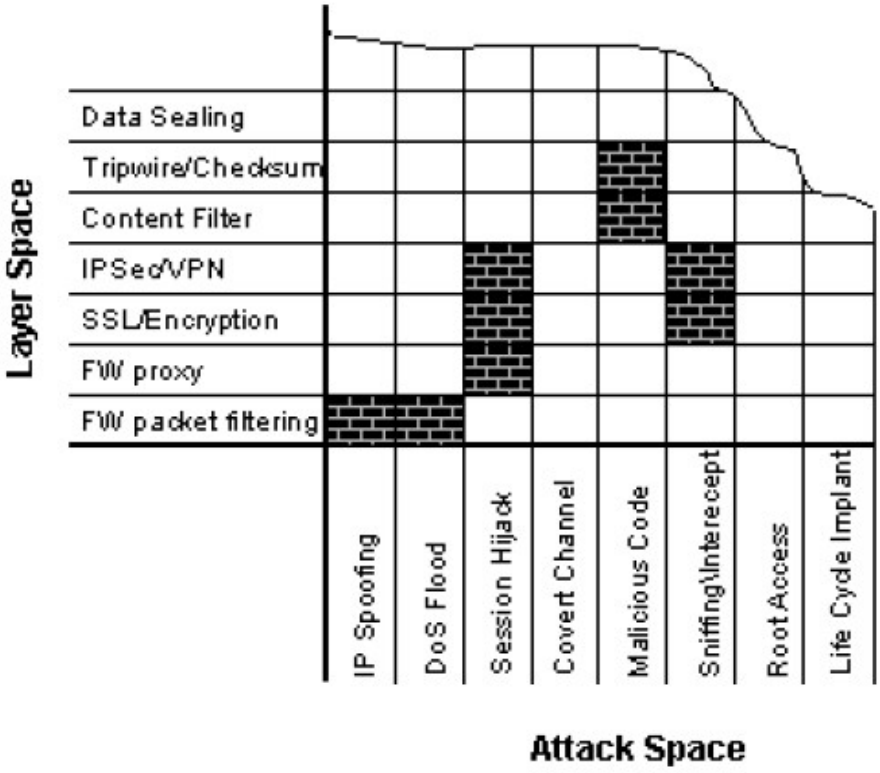

圖 7 深度防禦 vs 廣度防禦

為了阻止各種網路攻擊行為，傳統的方法主要是採用單一的、靜態的安全技術來防護。這些方法在真正的網路攻擊行為發生時，尤其在遭受新型的網路攻擊方法攻擊時，可能會給系統帶來不可預期的損失。因此需要研究一些積極主動的網路安全防禦手法和反擊手法，並和傳統的網路防禦方法相結合，構建一個安全的網絡環境。在 2005 年的 2 月解放軍理工大學的周海剛等人根據網路主動防禦安全模型和縱深防禦策略，提出一種網路主動防禦系統體系結構[16]。如圖 8 所示。

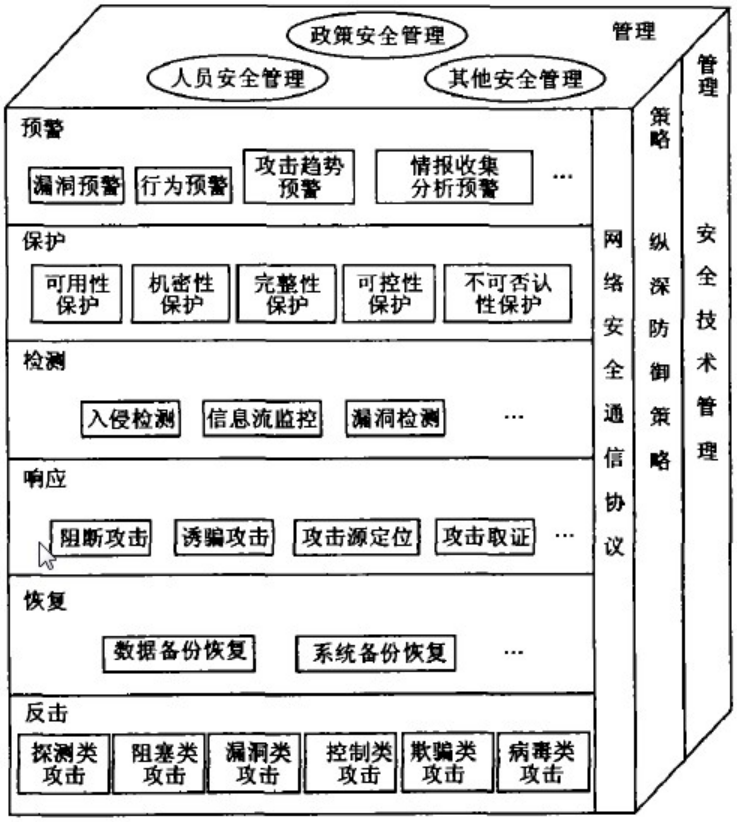

圖 8 虛網路主動防禦系統體系結構示意圖

網路主動防禦系統體系結構是一個三維的立體結構，分為 3 個層面：技術層面、策略和安全技術管理層面、管理層面。技術層面分為六層和一個網路安全通訊協定。這 6 層分別是：預警、保護、檢測、反應、恢復和反擊。策略和安全技術管理層面包括縱深防禦策略和安全技術管理。安全技術管理對這六層的技術進行管理，縱深防禦策略使得這六層的技術在統一的安全策略下能協調工作，來共同構築一個多層縱深的防護體系。而這篇論文最大的特色在於其坐了提供一般預警、保護、檢測等功能外，還提出反擊這樣的概念，反擊指的是網路反向攻擊，其綜合運用各種網路攻擊手法對攻擊者進行反攻擊，迫使其停止攻擊。這些攻擊手段包括探測類攻擊、阻塞類攻擊、漏洞類攻擊、控制類攻擊、欺騙類攻擊和病毒類攻擊等。

不過本研究的目的在於提出一個可行的網路安全防護機制，而非進行攻擊機制之探討，因此這一個新穎的概念在本研究中並未採行。

3.2. 虛擬化技術之相關研究

現今虛擬化技術的各方面都有了進步，虛擬化技術也從純軟體虛擬化逐漸演進到處理器級虛擬化、平台級虛擬化及 I/O 虛擬化，在 x86 平台上代表性技術就是 2006 年 Intel 公司所提出的處理器輔助虛擬化技術 Intel Virtualization Technology[2]，這是 x86 虛擬化的第一步，其分為 Itanium 平台的 VT-i 和對應 x86 平台的 VT-x 兩個版本。純軟體虛擬化主要的問題是效能和隔離性。完全虛擬化技術(Full Virtualization)可以提供較好的 Guest OS 隔離性，不過其效能不高，在不同的應用下，可以消耗掉主機 10%~30%的資源。而作業系統虛擬化技術(OS Virtualization)可以提供良好的效能，然而各個 Guest OS 之間的隔離性卻不強。無論是何種軟體方法，隔離性都是由 Hypervisor 層所提供的，過多的隔離會導致效能下降。而這些問題主要跟 x86 設計時並沒有考慮到虛擬化有關。x86 架構為了保護指令的運行，提供了 4 個不同 Privilege Level，術語稱為 Ring，從 Ring 0～Ring 3。Ring 0 的優先級最高，Ring 3 最低。各個 Privilege Level 對可以運行的指令有所限制，例如：GDT、IDT、LDT、TSS 等指令就只能運行於 Ring 0。

作業系統必須要運行一些 Ring 0 的特權指令，因此 Ring 0 是被用於運行作業系統核心，Ring 1 和 Ring 2 是用於作業系統服務，Ring 3 則是用於應用程式。然而實際上並沒有必要用完 4 個不同的 Privilege Level，一般的作業系統實作上都只使用兩個 Privilege，即 Ring 0 和 Ring 3，如圖 9 所示：

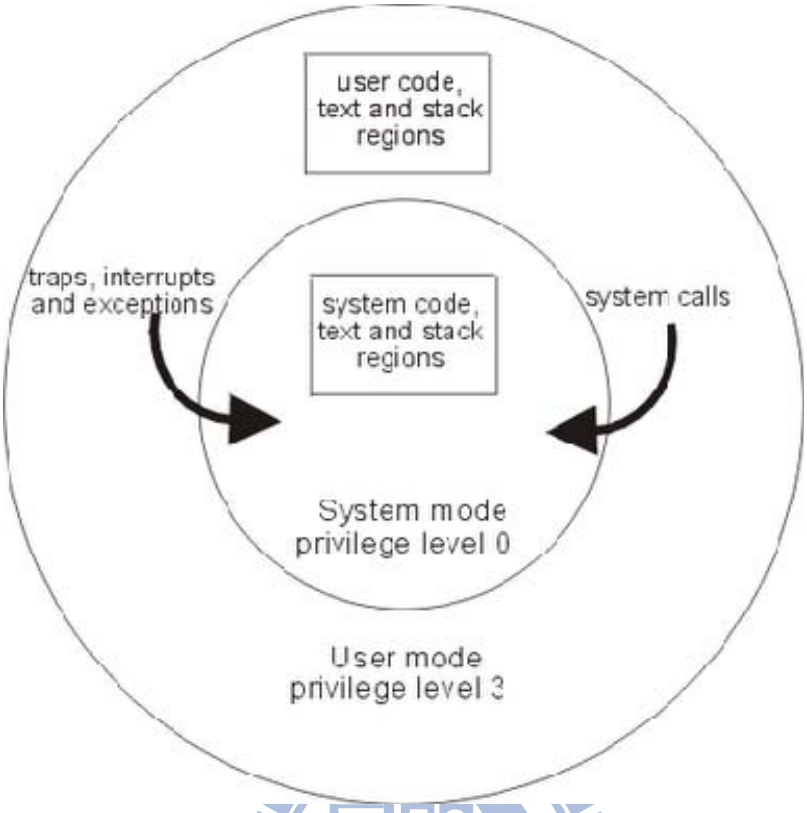

圖 9 Privilege Level 示意圖

也就是說在一個 x86 作業系統中，系統核心必須運行於 Ring 0，而 VMM(Virtual Machine Monitor，虛擬機監視器)以及其管理下的 Guest OS 却不能運行於 Ring 0，因為那樣就無法對所有虛擬機器進行有效的管理。在没有處理器輔助的虛擬化情況下，挑戰就是採用 Ring 0 之外的等級來運行 VMM、Hypervisor 以及 Guest OS。但無論使用何種等級來運行，Guest OS 還是無法運行於 Ring 0，則特權指令就必須通過模擬的方式來運行，這會帶來很明顯的效能問題。

同時這些特權指令如果隔離不當可能嚴重威脅到其他 Guest OS 甚至 Host OS。Ring Deprivileging 技術使用 IA32 架構的 Segment Limit 模式和 Paging 模式來隔離 VMM 和 Guest OS，但 64 位元的作業系統並不支援 Segment Limit 模式，因此要運行 64 位元作業系統，就必須使用 Paging 模式。對於虛擬化而言，Paging 模式無法區分 Privilege

0/1/2 模式，因此 Guest OS 一定要運行在 Ring 3，這樣 Paging 模式才可以將 Host OS 和 Guest OS 隔離開來，然而在同一個 Privilege 模式下的不同應用程式，例如：不同的虛擬機器是無法受到 Privilege 機制所保護的，這就是目前 IA32 帶來的隔離性問題，這個問題被稱為 Ring Compression。

因此 Intel 公司提出了 Intel Virtualization Technology 來提升虛擬化效率和虛擬機器的安全性，一般稱 IA32 平台為 VT-x，而在 Itanium 平台上的 VT 技術則稱為 VT-I，其架構如圖 10 所示。

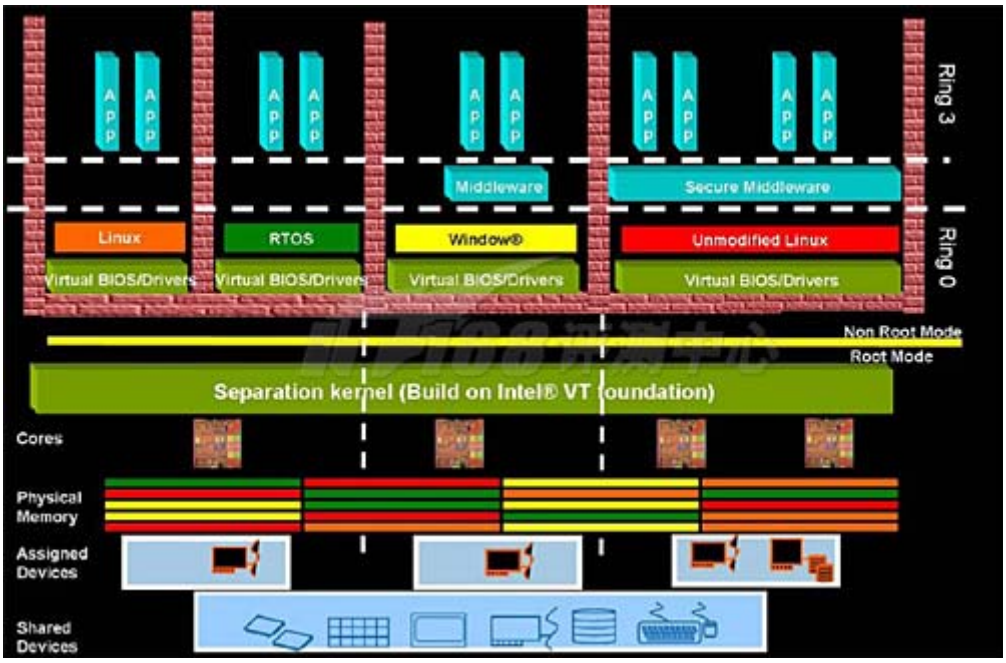

圖 10 Intel Vitrualization Technology 虛擬化架構示意圖

VT-x 將 IA32 的 CPU 運作分成兩層：VMX root operation 和 VMX non-root operation，VMX root operation 設計給 VMM/Hypervisor 使用，而 VMX non-root operation 則是另一個處在 VMM 控制之下的 IA32 環境。每一層都支援四個 Privilege Levels，這樣在 VMX non-root operation 環境下運行的虛擬機器就能完全地利用 Ring 0 等級。Intel VT 同時為 VMM 和 Guest OS 提供了所有的 Privilege Levels，而不是只

讓它們分別占據一個 Privilege Level，因為 VMM 和 Guest OS 運行在不同的兩個環境。

藉由這樣的設計，以往需要透過模擬運行的特權指令就能正常地運行於虛擬機器內部了。而 VMM 也能從模擬運行特權指令當中解放出來，這樣既能解決 Ring Aliasing 問題（軟體運行的實際 Ring 與設計運行的 Ring 不相同帶來的問題），又能解決 Ring Compression 問題，從而大大地提升效率。

為了建立這種兩個虛擬化窗體的架構，VT-x 設計了一個 Virtual-Machine Control Structure（VMCS，虛擬機器控制結構）的資料結構，包括了 Guest-State Area 和 Host-State Area，用來保存虛擬機器以及主機的各種狀態參數，並提供了 VM entry 和 VM exit 兩種操作在虛擬機器與 VMM 之間切換，用户可以通過在 VMCS 的 VM-execution control fields 裡面指定在執行何種指令或發生何種事件的時候，VMX non-root operation 環境下的虛擬機器就執行 VM exit，從而讓 VMM 獲得控制權，因此 VT-x 解決了虛擬機器的隔離問題，又解決了效能問題。

Inter VT 的出現解決了重要的虛擬處理器架構問題，讓純軟體虛擬化解決方案的效能問題得以大大緩解。但對於效能之影響不僅只有處理器，I/O 也是很重要的一部份，因此無論是儲存、網路或是顯示卡等，I/O 能力都是企業虛擬化架構的一個重要部分。隨著整體處理器資源的利用效率的提升，對資料的 I/O 處理能力也提出了更高的要求。

當時的 I/O 設備虛擬化主要是利用模擬方式，因此效能上很容易成為瓶頸，因此在 2007 年 5 月 Intel 公司就適時提出了 Intel Virtualization Technology for Directed I/O 的概念[3]，簡稱為 Intel VT-d，其架構如

圖 11 所示。

Intel Direct I/O 虛擬化的關鍵在於解決 I/O 設備與虛擬機器資料交換的問題，而這部分主要相關的是 DMA 直接記憶體存取以及 IRQ 中斷請求，只要解決好這兩個方面的隔離、保護以及效能問題，就能夠成功的將 I/O 虛擬化。

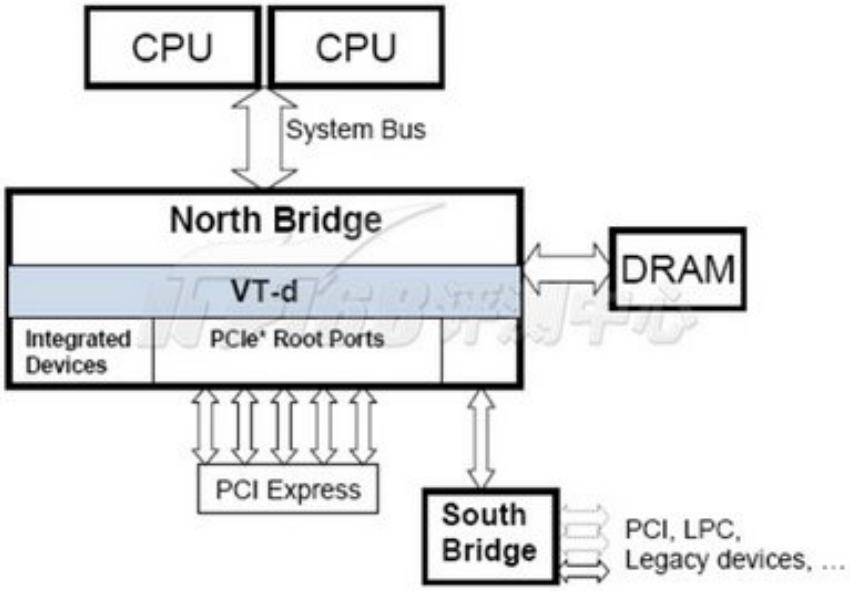

圖 11 Intel Direct I/O 虛擬化架構示意圖

和處理器上的 Intel VT-i 和 VT-x 一樣，Intel VT-d 技術是一種基於北橋晶片(North Bridge Chip)的硬體輔助虛擬化技術，透過在北橋晶片中內置提供 DMA 虛擬化和 IRQ 虛擬化硬體，提供新型的 I/O 虛擬化方式，Intel VT-d 能夠在虛擬環境中大大地提升 I/O 的可靠性、靈活性與效能。

I/O 設備會產生很多的中斷請求，I/O 虛擬化必須要能分離這些中斷請求，並正確傳送到不同的虛擬機器上。傳統設備的中斷請求可以具有兩種方式：一種是透過 I/O 中斷控制器來送，另一種是透過 DMA 寫入請求直接發送出去，由於需要在 DMA 請求內嵌入目標記憶體位

址，因此這個架構必須要能完全存取所有的記憶體位址。VT-d 進行的改動還有很多，如硬體緩衝、位址轉換等，透過 VT-d 技術，虛擬機器得以直接使用 I/O 設備或者 I/O 設備共享方式來代替傳統的設備模擬的方式，從而大大提升了虛擬化的 I/O 效能。

3.3. 網路效能分析之相關研究

在企業評估網路設備的流程中，實際測試佔了非常重要的一環，目的是為了找出符合需求的設備，避免 IT 投資的浪費。一般來說，網路設備的測試有兩種常見的做法，一種是將設備直接部署上線，觀察它在真實流量下的運作情況，至於另外一種方式則是利用工具取得數據。而透過工具測試網路設備，經常會與實際環境下的結果有所落差，不過對於測試時間有限的企業來說，仍然是可行的做法，而我們所要使用的工具是 Rutgers 大學的 Chung-Hsing Hsu 及 Ulrich Kremer 於 1998 年所發表的 Iperf[6]。網路吞吐量效能測試工具及 HP Rseach Lab 的 David Mosberger 及 Tai Jin 於 1998 年所發表的 httperf[7]網站效能測試工具，Iperf 可以產生 TCP 及 UDP 流量，協助測試網路吞吐量，而 httperf 則可以產生 http request 封包，藉由不同的參數設定，可用來測試網站之效能。這二套常用的免費網路效能測試工具，可以幫助企業測試網路架構的傳輸效能，或者找出實際環境當中的效能瓶頸。

在 2009 年 Y Koh 等人在 IEEE 期刊發表一篇利用委託的網路處理機制改善虛擬機器 Windows 作業系統網路效能的論文[4]，其所發表的**改善 Windows 虛擬機器網路效能的委託機制**-Linsock 其主要架構如圖 12 所示。

其主要是由 Winsock-Linsock Translator(簡稱 WLT)及 Linsock Server 所組成，WLT 安裝在 Guest OS 上，負責攔截 Winsock API 的

function call，Linsock Server 則安裝在 Host OS 上，負責透過 Host OS 上的 BSD sockets 來處理 Guest OS 的應用程式請求，並調整 Host OS 上的網路堆疊，而 WLT 和 Linsock server 都是跑在 Guest OS 及 Host OS 的 User-Level 的層級，並沒有修改其核心程式。

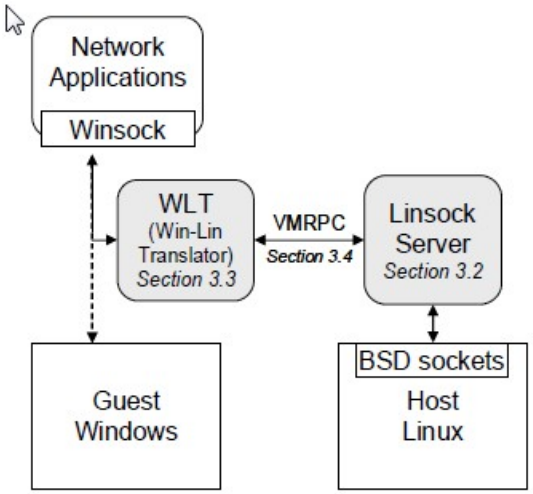

圖 12 Linsock 委託機制

為了說明 Linsock 委託機制效能比裝置模擬機制及半虛擬化機制要好，其利用 iperf 對這三種機制進行最大網路吞吐量效能測試，如圖 13 所示。

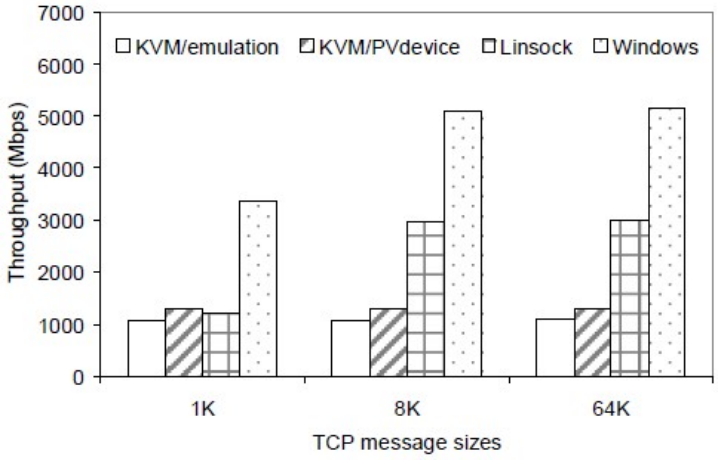

圖 13 Iperf 最大網路吞吐量比較圖

圖 13 可以看到在封包長度只有 1K 時，三種機制的最大網路吞吐

量差不多，但在封包長度為 8K 及 64K 時，Linsock 委託機制的網路吞吐量效能只和利用實際主機架構的 windows 差 40%，但也比其他兩種機制高出 30%左右。

該篇論文還有進行虛擬機器之間利用不同機制所進行的測試，如圖 14 所示，利用裝置模擬機制的效能非常差，在 10G 的網路下最大吞吐量竟然只有 100Mbps 左右，而半虛擬化機制約有 550Mbps 左右，而 Linsock 機制在封包長度 1K 時約有 730Mbps 左右，在封包長度 8K 及 64K 時卻有 2.2Gbps 左右。

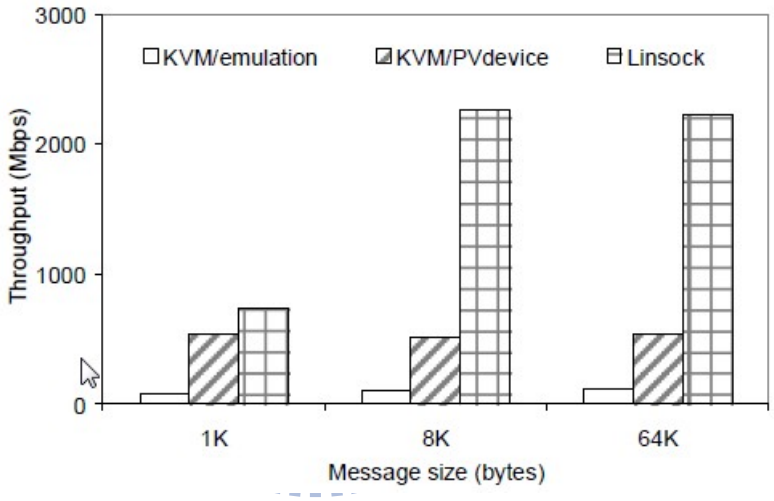

圖 14 Inter-VM 網路吞吐量效能測試比較圖

而這篇研究為了要測試實際網路應用程式的效能如何，還架設了 Apache Web Server，並在上面放置一個 128K 的檔案，再利用 httperf 對這些機制進行回應時間效能測試比較，如圖 15 所示，可以看出利用 Linsock 機制，網頁伺服器的回應時間比其它兩種機制要低，其回應時間約在 2.1ms 左右，而裝置模擬機制約為 10.3ms，半虛擬化機制約為 6.5ms 左右。

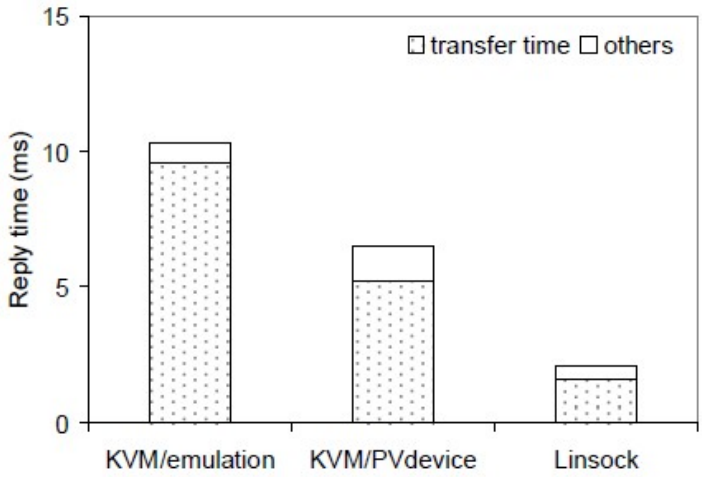

圖 15 Httperf 回應時間比較圖

本研究把 3.1 節及 3.2 節相關研究的特點整合，發展出一套虛擬化技術之縱深防禦架構，將於第四章進行完整介紹，並利用 3.3 節介紹的 iperf 及 httperf 等網路效能測試工具進行網路效能分析，將其測試過程及分析結果於第五章進行完整說明。

第四章　植基於虛擬化技術之網路安全縱深防禦架構設計(Network Security Defense-in-Depth Architecture Design based on Virtualization Technology)

隨著網際網路的持續發燒、駭客入侵或破壞網站事件的頻傳，網路安全防護不單是企業界急於解決的重要難題，還希望防護成本能夠下降，因此本研究希望藉由提出運用虛擬化技術將防火牆、入侵防禦系統以及防毒牆等安全防護技術整合之構想，能達到有效加強網路系統之安全防護之能力、增加駭客攻擊成本與改善企業資源浪費的問題，使網路安全解決方案可臻至完美。

本章 4.1 節首先概述本研究之虛擬化網路安全縱深防禦架構之設計，並於 4.2 節細部介紹各功能模組的作用，接著在 4.3 節說明本研究建置方式以及建置過程中之注意事項說明，最後於 4.4 節描述虛擬化技術所建置的縱深防禦架構所達到應有的防護效果。

4.1. 虛擬化網路安全縱深防禦架構概述

本研究將進行虛擬化縱深防禦架構之設計，設計之層級架構如圖 16 所示，本研究將虛擬化網路安全縱深防禦架構分為四層，每一層的特性如下：

實體層(Physical Layer)：

實際的硬體資源，如 CPU、記憶體、網路卡、磁碟機等。

監督層(Hypervisor Layer)：

支援虛理器虛擬化的虛擬機器管理員(Virtual Machine

Management，簡稱 VMM)，其在 Guest 作業系統及硬體層之間提供一個抽象層，在這個抽象層上允許任何作業系統在硬體上執行。

虛擬網路層(Virtual Network Layer)：

其網路模組主要是模擬虛擬交換器及調整虛擬機器網路架構。

安全層(Security Layer)：

其安全模組主要是針對網路闡道端流入流出的封包進行檢測。

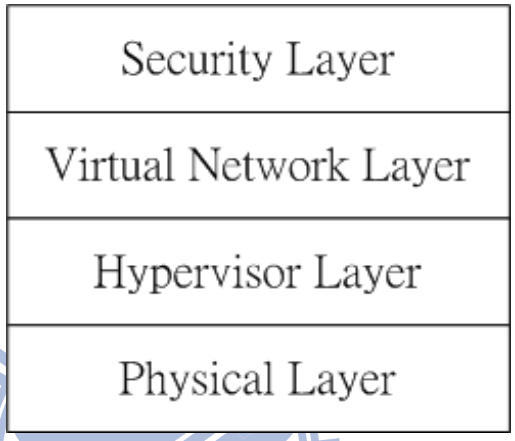

圖 16 虛擬化網路安全縱深防禦架構

圖 17 為虛擬化網路安全縱深防禦架構功能模組圖，其由三大功能模組所組成，功能如下所示：

管理模組(Management Module)：

協調硬體層的計算資源及控制各虛擬機器之 CPU、記憶體、磁碟空間、網路卡等資源分配比例。

網路模組(Network Module)：

管理虛擬機器之間封包交換的路徑，能夠準確的控制在虛擬機器之間封包行經的路徑。

安全模組(Security Module)：

過濾及檢測進出的封包是否有惡意連線或行為，做進一步的存取控管機制，本研究所提的防禦機制皆部署於此。

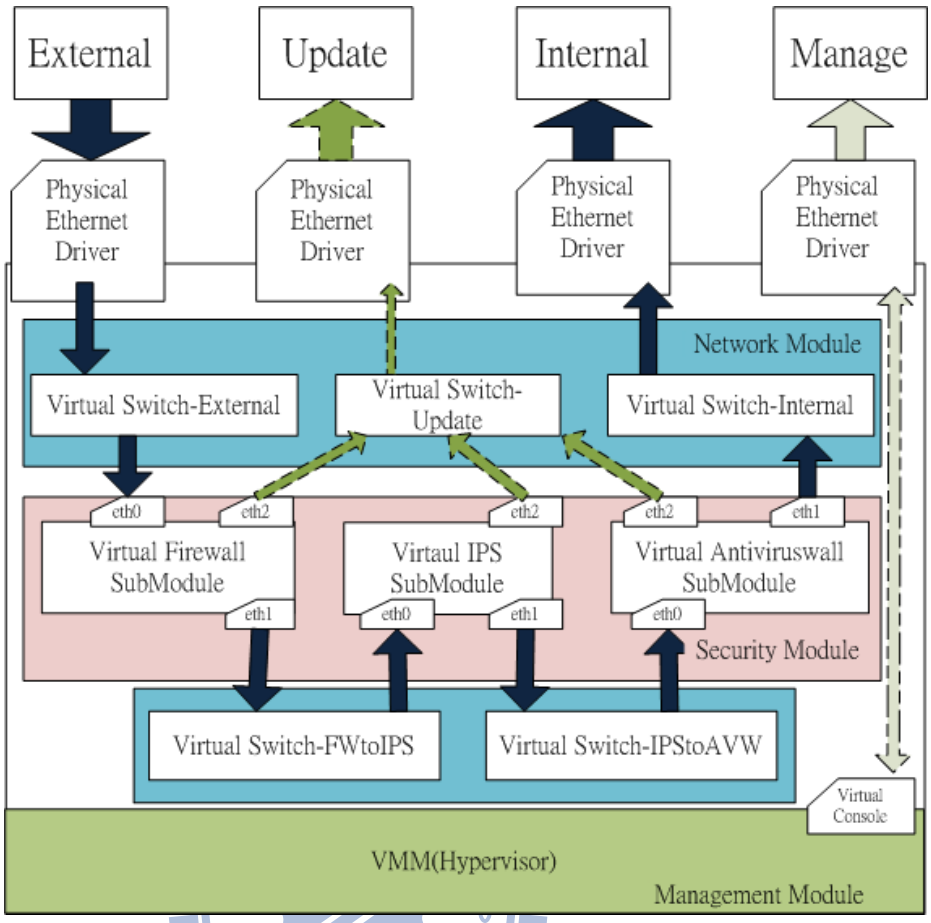

圖 17 虛擬化網路安全縱深防禦架構模組圖

本研究將於4.2節更細部的說明虛擬化網路安全縱深防禦架構中所有模組之功能。

4.2. 虛擬化網路安全縱深防禦架構功能模組

虛擬化網路安全縱深防禦架構除了有 4.1 節概述的三大功能模組外，在安全模組中，本研究設計了三套針對不同防禦性質的子功能模組，並且利用網路模組將其封包傳遞路徑定義清楚，底下我們將介紹各功能模組之主要功能。

(1)管理模組(Management Module)

協調硬體層的計算資源及控制各虛擬機器之 CPU、記憶體、磁碟空間、網路卡等資源分配比例。例如：可以針對不同防禦主機給於適

當之計算資源，充份有效利用有限之硬體資源，而其管理模組不會佔用過多計算資源，且能夠提供高可用性、高擴充性及高集中性的特性，管理者可透過遠端終端機利用 VPN 技術連線至 VMM 對虛擬機器進行管理，並且提供帳號驗證之功能，除了提高安全性外，也提高便利性。

(2)網路模組(Network Module)

在網路模組方面，有提供虛擬網路卡及虛擬交換器之模組，透過虛擬網路卡，可使各虛擬機器擁有良好的擴展能力，而虛擬交換器則能提供虛擬機器的切割及統一的封包傳遞路徑，另外還有提供對外線上更新及遠端終端機的管理功能。

本研究中虛擬機器之網路架構必須建立 5 台虛擬交換器(Virtual Switch)，這 5 台虛擬交換器之功能如表 2 所示。

表 2 虛擬交換器用途一覽表

虛擬交換器名稱	用途
Internal	連接內部網路之網段
External	連接外部網路之網段
FWtoIPS	連接防火牆和入侵防禦系統之通道
IPStoAVW	連接入侵防禦系統及防毒牆之通道
Update	軟體更新

(3)安全模組(Security Module)

安全模組可提供過濾檢測進出之封包的能力，這一部份可以實作非常多種防禦機制，本研究將網路安全縱深防禦機制部署於此，並且提供三種不同性質之防禦機制進行實作，圖 18 為本研究封包傳遞的路徑，當封包進到防火牆時，會有一些不允許的非法連線封包被被阻擋，而通過的封包皆為合法的連線，當封包經過入侵防禦系統時，會發現

合法連線中的非法行為，以網頁伺服器為例，如果有 Cross-Site Scripting 的網頁攻擊行為發生，在入侵防禦系統就會把它阻擋下來，但如果封包是由使用者端所下載的惡意程式，那就入侵防禦系統就無法進行阻擋，這時會透過防毒牆將其阻擋下來，進行封包重組，檔案過濾後，將其刪除，讓真正正常的網路封包流進內部網路。底下將介紹各個防護機制的檢測流程。

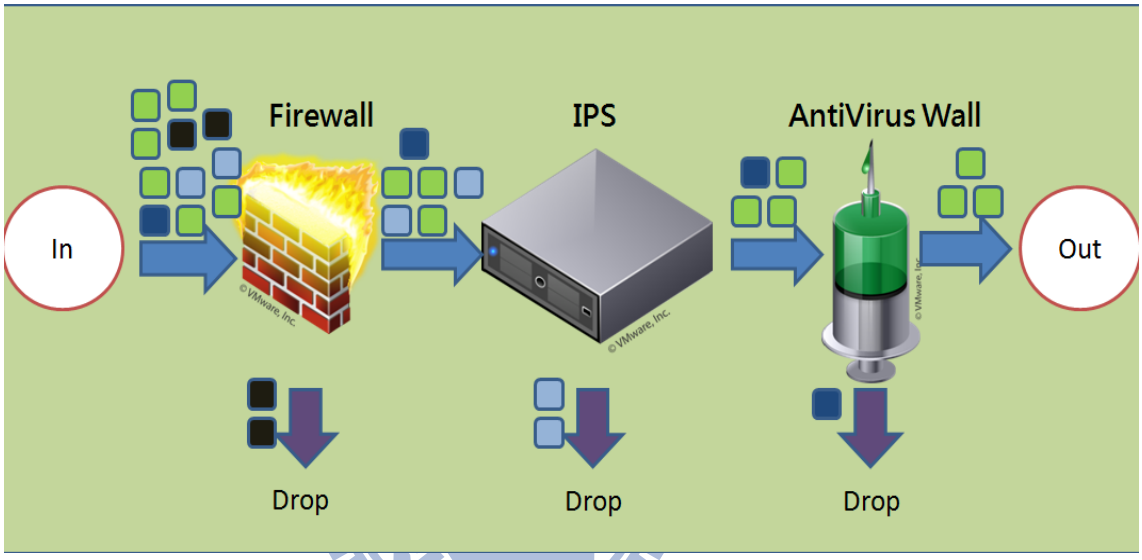

圖 18 虛擬化縱深防禦網路封包流程示意圖

(1)虛擬化防火牆系統子模組(Virtual Firwall SubModule)

在虛擬化網路安全縱深防禦架構中，本研究使用 Netfilter/Iptables 這套頗富盛名的開放原始碼防火牆，Netfilter 是在 Linux 核心(Kernel)的模組，它只會依照管理員訂下的規則，對網路封包進行過濾，並且依靠 Iptables 應用程式將規則傳至 Netfilter 模組。

防火牆是透過一系列的規則而建成，而每個規則都包含了一個對封包的描述(Match)和一個處置動作(Target)。每當封包符合規則中的描述時，核心便會對封包進行相應的處置動作。而在 Netfilter 的角度中，這些規則是記錄在不同的鏈(Chain)中，而鏈又會被歸納到不同的規則

表(Table)中。封包會根據它在核心中不同的層次和狀態，被送到一個或多個規則表和鏈中，並與當中的每條規則進行比對並執行相對應的動作，在核心中有三個規則表，分別是 FILTER、NAT 及 MANGLE，在本研究中僅使用到 FILTER 及 NAT 兩個規則表，而這兩個規則表又有一些預設的鏈，在 FILTER 規則表中有 INPUT Chain (給目的地是本機的封包)、 FORWORD Chain(給途經本機的封包)、OUTPUT Chain (給由本機所發出的封包)。而在 NAT 規則表中有 PREROUTING Chain (給所有進入本機而尚未經過路由處理的封包)、POSTROUTING (給所有已經經過路由處理而目的地不是本機的封包)及 OUTPUT Chain，所以我們將這兩個規則表中的鏈整合起來如圖 19 所示。

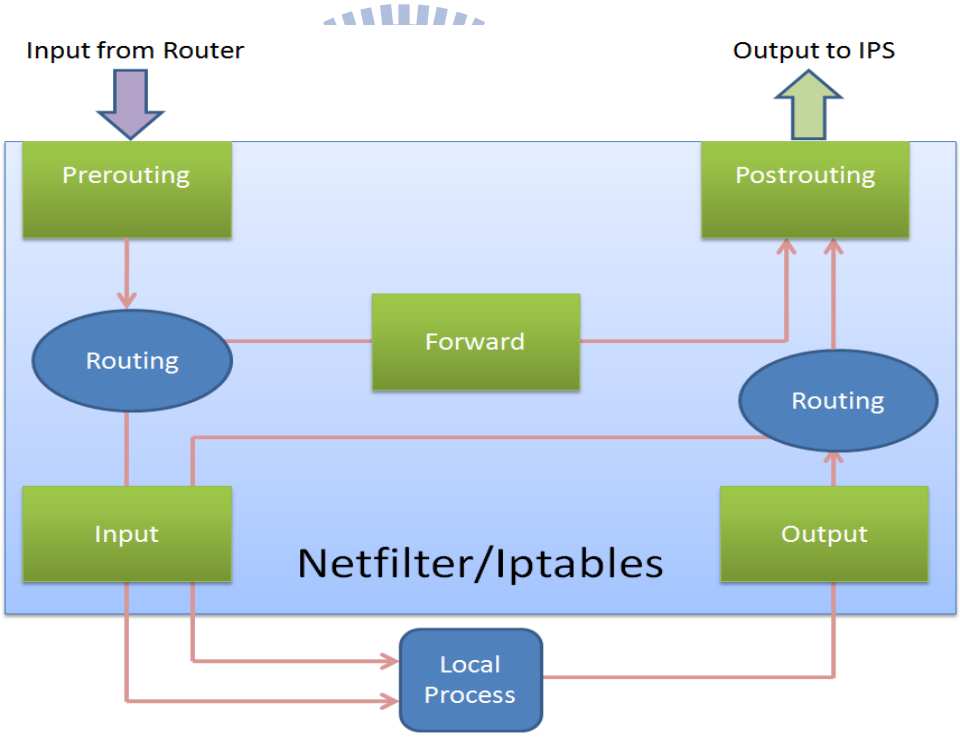

圖 19 防火牆內部運作流程圖

圖 19 是網路封包流進 Iptables 防火牆之流程圖，封包一進來會先進入 NAT 規則表，其主要是進行網路位址轉換 (Network Address Translation) 的功能。顧名思義網路位址轉換就是把封包標頭的位址和

通訊埠的資料更改。這個功能主要的好處是可以讓內網所有使用者共用同一個 IP 位址連上 Internet，且也可以隱蔽內路網路的拓撲，運用網路位址轉換就可把所有內部網路對外的通訊轉換成同一個 IP 位址所發出的封包一樣。NAT 轉換剛完成後，會進入路由表中選擇路徑，如果是往防火牆主機本地，就會被導向 INPUT Chain 中過濾，如果是要進入內部網路，則會導向 FORWARD Chain 中進行過濾，最後可被接受的封包都會從 Postrouting Chain 往入侵防禦系統送過去。

(2)虛擬化入侵防禦系統子模組(Virtual IPS SubModule)

在本研究的虛擬化網路安全縱深防禦架構中，所使用的入侵防禦系統是 Snort 這套頗具代表性的開放原始碼入侵偵測系統搭配 Iptables Queue 模組，這樣的搭配會將防火牆送進來的流量導向一個佇列中，再由 Snort 將佇列的資料逐筆進行過濾，非法的封包即將其丟棄，合法的封包即送往防毒牆。圖 20 顯示了封包進入入侵防禦系統後經過的流程。

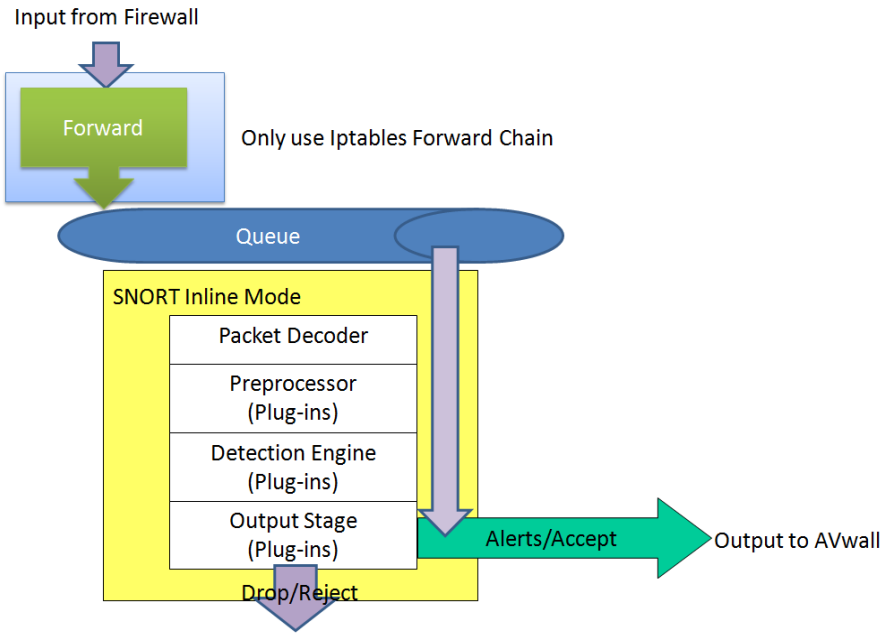

圖 20 入侵防禦系統內部運作流程圖

當網路封包進入入侵防禦系統時，會先進到 Iptables 的 Forward Chain(其他 Chain 預設為 Accept)，並且從這裡將封包導向佇列之中，而 Snort inline mode 會依序一筆一筆從 Queue 中讀取封包，並將封包解碼，接著進行特徵碼(signature)比對的作業，最後決定是否放行或是丟棄。放行後會往防毒牆的方向傳過去。

(3)虛擬化防毒牆系統子模組(Virtual Antiviruswall SubModule)

在本研究的虛擬化網路安全縱深防禦架構中，所使用的防毒牆是 ClamAV 這套在 Linux 底下非常有名的免費防毒軟體加上 HTTP Anti-Virus Proxy(HAVP)這套 HTTP 的防毒代理伺服器，並且 Iptables 要配合將封包重導向(Redirect)至本機 HAVP 所開啟的網路通訊埠 8080 進行封包重組及過濾，待利用 ClamAV 的病毒碼資料庫進行掃描後，將合法的檔案轉成封包重新傳送至用戶端電腦上，非法的檔案即將其丟棄。圖 21 顯示了封包進入入侵防禦系統後經過的流程。

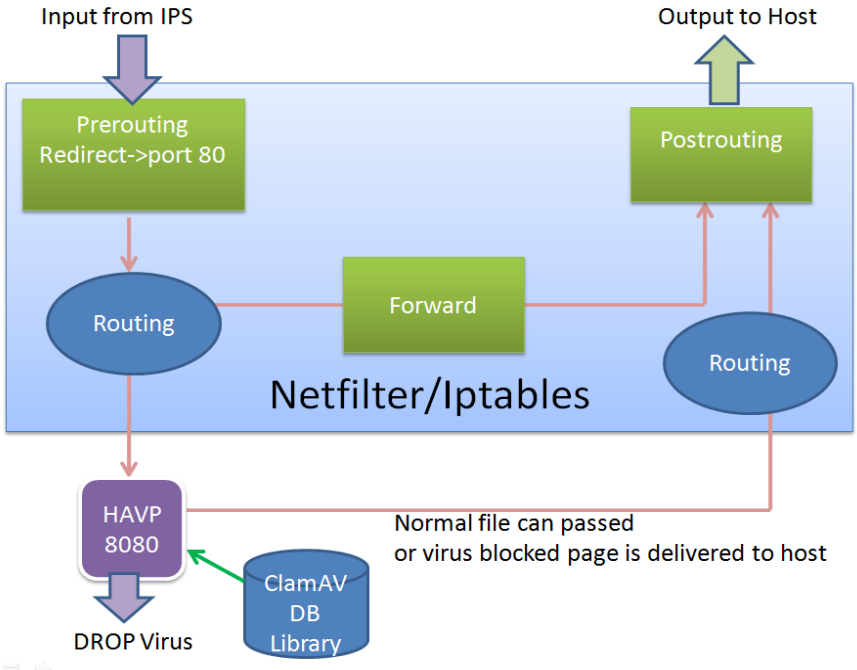

圖 21 防毒牆內部運作流程圖

從圖 21 可以看到，當封包進入 NAT 規則表中的 PREROUTING Chain 後，會直接將網路通訊埠為 80 的封包重導向至本機的網路通訊埠 8080，8080 埠是 HAVP 的掃描應用程式，經由 HAVP 透過 ClamAV 的病毒碼資料庫掃描完成後，正常的檔案會再經由路由表傳至使用者端電腦，而異常的檔案會被拒絕存取，且會回傳給使用者端電腦一個拒絕存取的網頁。

4.3. 虛擬化網路安全縱深防禦架構建置說明

本研究係利用 VMWare 虛擬化技術做為管理模組，並利用其內建的虛擬交換器做為網路模組，並以此建置出一個縱深防禦網路安全架構，透過虛擬平台管理員可對環境組態進行設定，並對各虛擬機器進行細項設定，例如：CPU 數量、記憶體大小、儲存空間及網路裝置等。整個虛擬化網路安全縱深防禦架構如圖 22 實驗環境配置圖所示。

圖 22 為本研究的實驗環境配置，本研究使用 VMWare ESX Server 4.1 作為此架構之虛擬平台，搭配實際硬體使用的規格如表 3 所示。

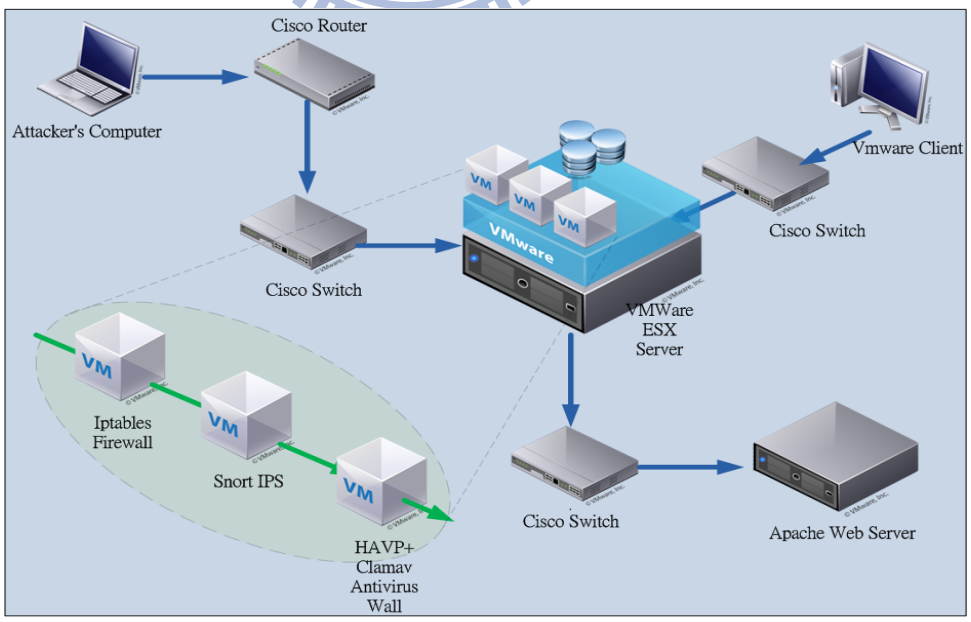

圖 22 實驗環境配置圖

本研究實驗環境之 VMWare ESX Server、VMWare Client、Virtual Machine 及外部實際的伺服器、交換器及路由器之硬體規格、作業系統及應用程式版本資訊如表 3 所示：

表 3 硬體設備與作業系統規格表

VMWare ESX Server		VMWare Client	
Model	IBM System X3250 M2	Model	HP EliteBook 6930p
CPU	Intel xeon 2.66GHZ Quad Core	CPU	Intel Core 2 Due T9550 2.67GHZ
RAM	8GB DDR2 800MHZ	RAM	2GB DDR2 800MHZ
OS	VMWare ESX Server 4.1	OS	Windowns 7 Enterprise x86
LAN	Intel PRO/1000 PT Dual	APP	VMWare vSphere Client 4.1
DISK	Segate 160G SAS x 2 (RAID0)	LAN	Intel 86567LM Gigabit NIC
		DISK	Segate 250G SATA
Virtual Machine(Firewall)		Virtual Machine(IPS)	
CPU	vCPU 2.66GHZ Single	CPU	vCPU 2.66GHZ Single
RAM	2 GB	RAM	2GB
OS	CentOS 5.5	OS	CentOS 5.5
Kernel	Kernel-2.6.18.194.32.1	Kernel	Kernel-2.6.18.194.32.1
APP	Iptables V1.3.5	APP	Iptables-1.3.5 Snort -2.8.6
LAN	vNIC(e1000) X3	LAN	vNIC(e1000) X3
DISK	20G SCSI	DISK	20G SCSI
Virtual Machine(AntiVirus Wall)		Web Server	
CPU	vCPU 2.66GHZ Single	CPU	Intel xeon 2.66GHZ Quad Core
RAM	2 GB	RAM	2GB DDR2 800MHZ
OS	CentOS 5.5	OS	CentOS 5.5

Kernel	Kernel-2.6.18.194.32.1	Kernel	Kernel-2.6.18.194.32.1
APP	Iptables-1.3.5 ClamAV-0.97 HAVP-0.92	APP	Apache-2.2.3-43
LAN	vNIC(e1000) X3	LAN	Intel PRO/1000 PT Dual
DISK	20G SCSI	DISK	Segate 160G SAS x 2 (RAID0)
Attacker's Computer		Cisco Switch	
CPU	Intel xeon 2.66GHZ Quad Core	Model	Catalyst 2950 Serials
RAM	2GB DDR2 800MHZ	IOS	12.2
OS	Windows XP SP3	PORT	12 Fastethernet ports
APP	N-Stalker 2009 Apache-2.2.3-43(eicar 2.0 virus_sample file) NMAP-5.50	Cisco Router	
		Model	Cisco-2600 Serials
		IOS	12.2
LAN	Intel PRO/1000 PT Dual	PORT	2 Fastethernet ports
DISK	Segate 160G SAS x 2 (RAID0)		

設定虛擬機器之網路架構

圖 23 為本研究之網路架構示意圖。由圖中可看到虛擬平台利用兩張 Intel 網路卡和外部網路對接，一張接在實體交換器上並設定為 vlan10，設定其為內路網路，而另一張接在外部網路並設定其為 vlan20，這樣實體交換器即可將封包送往 ESX 伺服器上的虛擬交換器 (Virtual Switch，vSwitch) 中，而根據 4.2 節的網路架構設計，在 VMWare ESX Server 中會有 5 部虛擬交換器，且每部虛擬化防護主機皆有 3 張虛擬網路卡，設定如圖 24 所示。

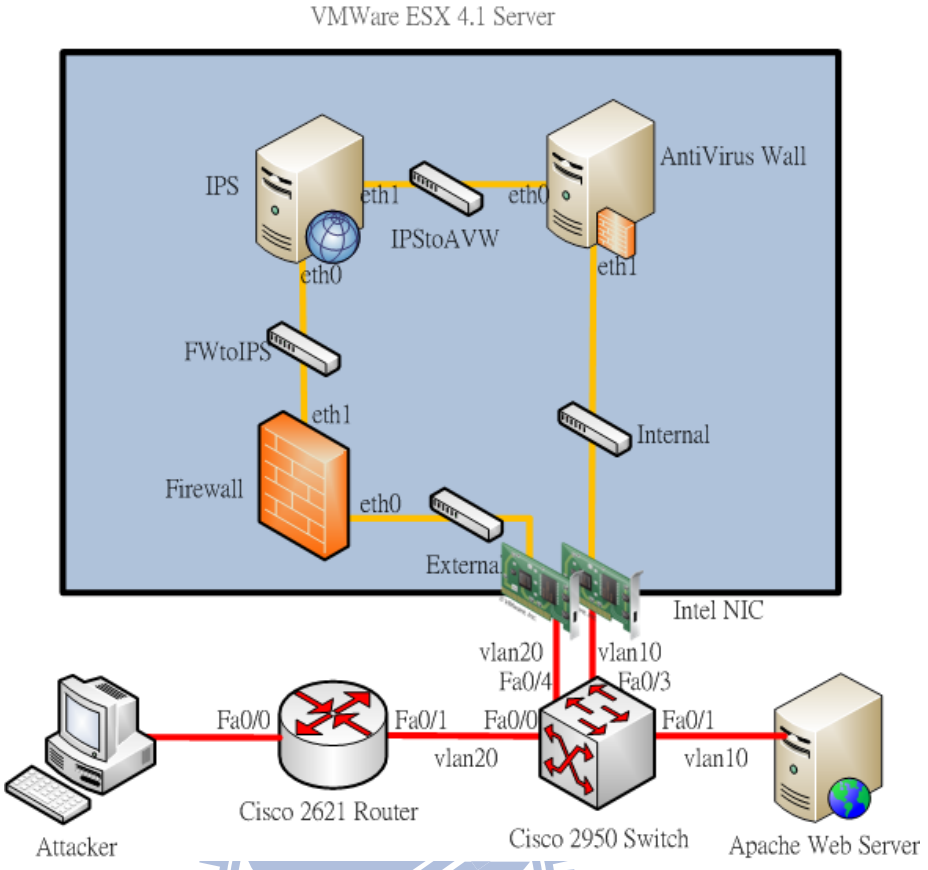

圖 23 虛擬化網路安全縱深防禦網路架構示意圖

如圖 24 所示，有 3 台虛擬交換器對外各自設定不同實體網路卡，vlan10 (Internal)虛擬交換器對外網路卡設定為 vmnic3，而 vlan20 (External)虛擬交換器對外網路卡設定為 vmnic0，而 ADSL (Update)接在 vmnic2 的實體網路卡上，另外有兩台虛擬交換器不用設置對外網路卡，其主要目的是給虛擬防護主機轉送網路封包到另一台虛擬防護主機上。

本研究的網路邏輯接線規則為將防火牆虛擬機器之一張虛擬網路卡(vNIC)接在 vlan20 虛擬交換器上，另一張虛擬網路卡接在 vlan30 虛擬交換器，而在入侵防禦系統虛擬機器中，將一張虛擬網路卡接在 vlan30 虛擬交換器上，另一張虛擬網路卡接在 vlan40 虛擬交換器，最後在防毒牆虛擬機器中將一張虛擬網路卡接在 vlan40 虛擬交換器上，

另一張虛擬網路卡接在 vlan10 虛擬交換器上，這樣就能夠確認所有的網路封包會根據 4.1 節所設計出來的虛擬化網路安全縱深防禦架構所限定的路徑傳送，不會有跳過某一種防護機制的風險存在，另外我們將在各虛擬機器中保留一張虛擬網路卡做 ADSL 上網使用，主要目的為更新軟體版本以及下載所需要之安裝套件。

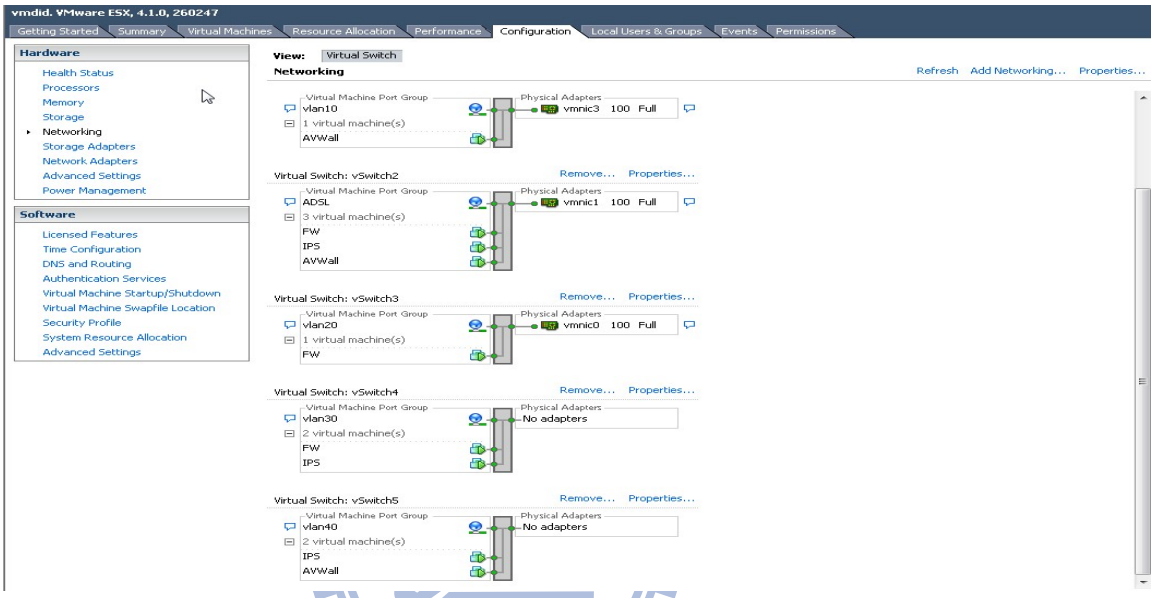

圖 24 虛擬交換器上 vlan 之設置

而每台虛擬交換器上都要開啟雜亂模式(Promiscuous Mode)，如圖 25 所示，因為 VMWare ESX Server 很嚴格規定虛擬交換器上 vlan 之間不能互通，這會讓本研究在虛擬機器上兩張不同 vlan 的虛擬網路卡設置為橋接模式(Bridge Mode)時，導致封包無法進行轉送，因此虛擬交換器上一定要開啟雜亂模式，封包才會依本研究所要求之路徑進行轉送(forward)的動作。

另外在配置虛擬網路卡時要注意，要選擇 VMWare 公司所提供之 e1000 虛擬網路卡，如果使用預設的虛擬網路卡，將造成網路效能低落之問題，VMWare 公司針對 Windows-based 作業系統有提供 vmxnet2 及 vmxnet3 兩種虛擬網路卡，使得網路效能得以提升，而在

Linux-based 的作業系統下提供 e1000 虛擬網路卡，以提升網路效能。

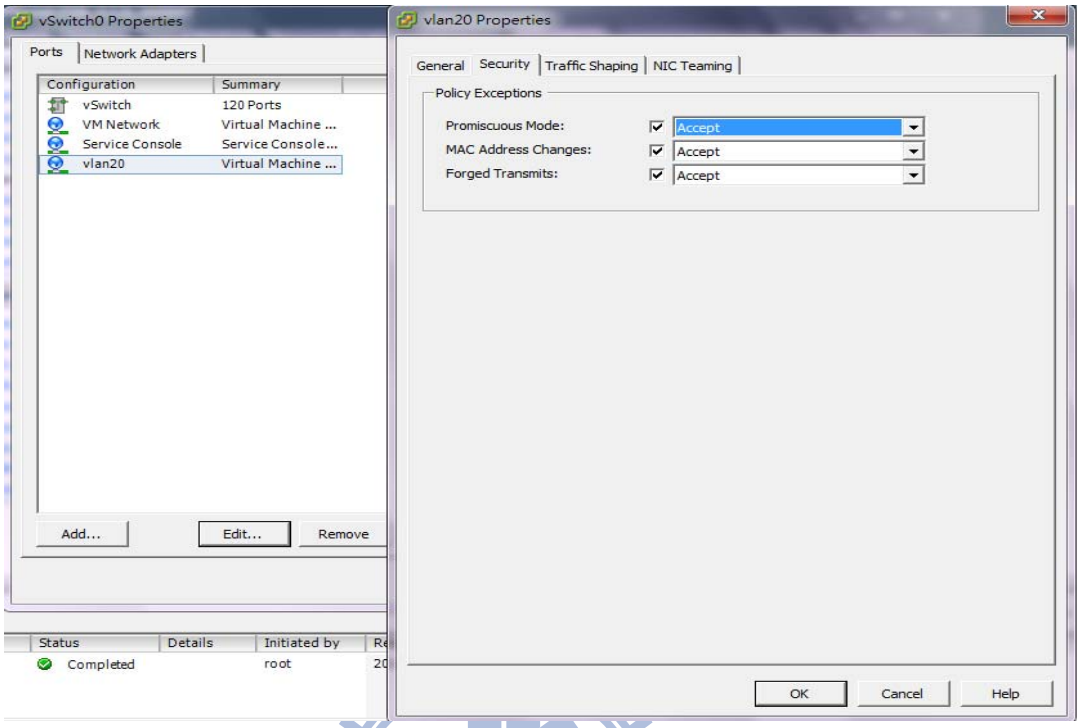

圖 25 在虛擬交換器中的 vlan 開啟雜亂模式

在實體的 Cisco 交換器上須將本研究中攻擊者網段設置為 vlan20(路由器連接到交換器上的連接埠)，而將內部的網頁伺服器所連接的網段設置為 vlan10(網頁伺服器的網路卡連接到交換器上的連接埠)，並且將 ESX Server 的兩張實體網路卡接在實體交換器上，並在實體交換器上設定其分別為 vlan10 及 vlan20，如此實體 Cisco 交換器才能和虛擬交換器交換封包。

虛擬機器之配置

當實體環境建置完成後，開始進行虛擬機器之配置說明，在本研究的實驗中須利用 VMWare vSphere Clinet 連線至 VMWare ESX Server，並在其中建立三個虛擬機器(Virtual Machine)，如圖 26 所示。

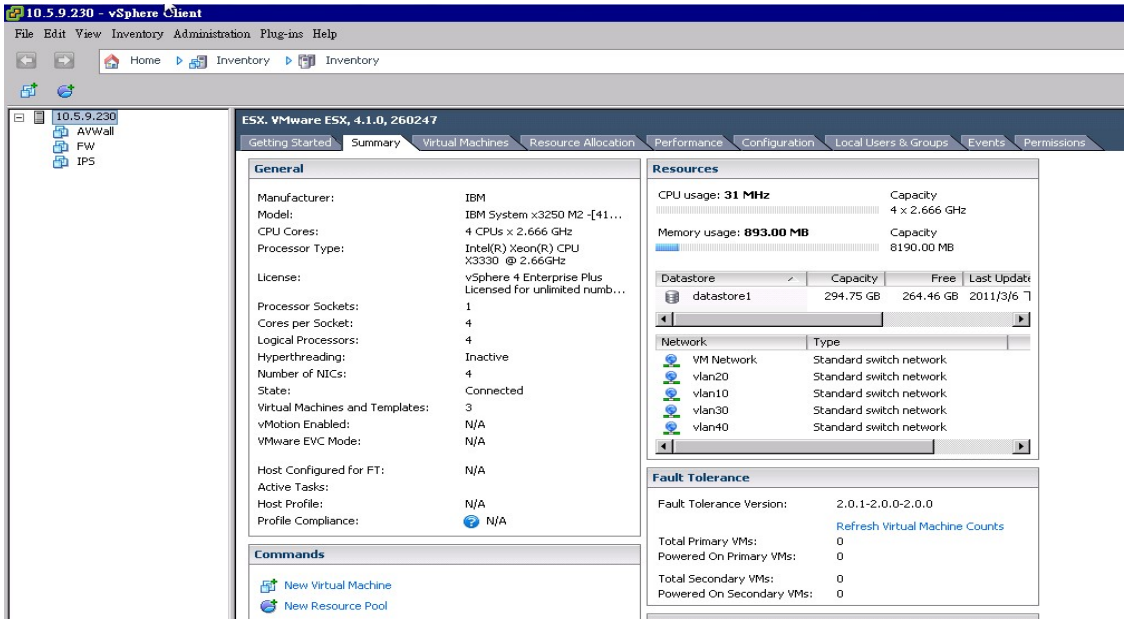

圖 26 在 VMWare ESX Server 中的 3 台虛擬機器

虛擬機器之作業系統安裝注意事項

本研究在虛擬機器上安裝開放原始碼的作業系統 CentOS 5.5，安裝時僅需注意網路卡不要使用 DHCP，並且將 IPv6 功能關閉，而在安裝完成後的第一次設定中，記得將預設防火牆及 SELinux 等存取控制功能關閉，以降低對虛擬化網路安全縱深防禦架構之影響。在安裝完作業系統後，可使用底下指令進行升級作業：

```
#yum –y update
```

或是利用底下指令安裝本研究所需要之套件：

```
#yum –y install package_name
```

例如：本實驗中我們在入侵防禦系統及防毒牆系統中將使用兩張網路卡實作橋接模式之網路架構，因此需執行底下指令安裝 bridge-utils 套件：

```
# yum –y install bridge-utils
```

如此即可使用 brctl 指令來將兩張網路卡進行橋接模式設定。如下所示：

```
# brctl addbr br0
# brctl addif br0 eth0
# brctl addif br0 eth1
# ifconfig br0 up
```

虛擬機器橋接模式之配置

不過在虛擬機器上每次都要利用上述指令進行網路卡橋接模式之設定較為不便，因此我們將利用自動橋接模式之設定方式，如下所示：

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
DEVICE=eth0 (另一張 eth1 網路卡也一樣的設定)
HWADDR=xx:xx:xx:xx:xx:xxv (每張網路卡皆不同，不用修改)
ONBOOT=yes
BRIDGE=br0
```

```
# vi /etc/sysconfig/network-scripts/ifcfg-br0
```

```
DEVICE=br0
TYPE=Bridge
BOOTPROTO=static
ONBOOT=yes
```

設定完成後重新開機或使用底下指令即可完成橋接網路卡的設定。

```
# /etc/init.d/network restart
```

防火牆建置要點

在防火牆建置部份，除將系統應用程式更新至最新版本外，還需注意必須開啟 ip_forward 功能，封包才會進行轉送。

```
# echo “1” > /proc/sys/net/ipv4/ip_forward
```

或修改/etc/sysctl.conf

```
#vi /etc/sysctl.conf
```

```
net.ipv4.ip_forward=1
```

存檔後執行 sysctl –p 即生效。

防火牆規則之部份，我們將防火牆規則分為三個部份，一個是預設防火牆規則及 NAT 設定的設定檔，一個是允許存取之設定檔，最後一個是不允許存取之設定檔，設定檔設定資訊如附錄一所示。

將附錄一之設定檔放置於 /usr/local/iptables/ 底下，並於 /etc/rc.d/rc.local 加上一筆/usr/local/iptables/iptables.rule，開機就會自動執行。

入侵防禦系統建置要點

入侵防禦系統[24]建置部份，需安裝的套件如下：

snort
libnet
iptables-devel
pcre-devel
glib2-devel
libpcap-devel
libnet
gcc
gcc-c++

在安裝 snort 前，先用 yum –y install 指令將 iptables-devel pcre-devel glib2-devel libpcap-devel gcc gcc-c+安裝好，接著將 libnet-1.0.2.tar.gz 解壓縮並進行編譯。

```
#tar zxvf libnet-1.0.2.tar.gz
#cd libnet-1.0.2
#./configure
#make && make install
```

安裝完成後，使用底下指令對 snort 原始套件(含主程式及規則檔)

進行編譯。

```
#tar zxvf snort-2.8.6.tar.gz
#cd snort-2.8.6.tar.gz
#./configure –enable-inline –dynamicplugin
#make && make install
#groupadd snort
#useradd –g snort:snort –s /sbin/nologin
#mkdir /etc/snort
#mkdir /etc/snort/rules
#mkdir /etc/snort/so_rules
#mkdir /var/log/snort
#chown snort:snort /var/log/snort
#cp ./snort-2.8.6/etc/* /etc/snort
#mkdir snortrule
#cd snortrule
#tar zxvf ../snortrules-snapshot-2860.tar.gz
#cp ./rules/* /etc/snort/rules
#cp ./so_rules/precompiled/CentOS-5.0/i386/2.8.6.0/* \
/etc/snort/so_rules
```

修改/etc/snort/snort.conf

```
#vi /etc/snort/snort.conf
```

```
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
output unified2: filename snort.log limit 128
```

snort 要運行 inline mode 必須搭配 iptables Queue 的功能才可進行過濾，須使用以下指令完成。

```
#modprobe ip_queue
```

#iptables –A FORWARD –i br0 –j QUEUE

上述設定完成後，使用底下指令啟動 snort

#/usr/local/bin/snort –D -Qvd –u snort –g snort –c \
/etc/snort/snort.conf –I br0 &

要在開機時啟動只要把上述指令加入/etc/rc.d/rc.local 中即可。

防毒牆建置要點

要先利用 yum –y install 安裝底下套件：

gcc

gcc-c++

zlib

zlib-devel

接著進行 ClamAV[26]防毒引擎之安裝：

#groupadd clamav && useradd -g clamav -M clamav

#tar zxvf clamav-0.97.tar.gz

#cd clamav-0.97

#./configure

#make && make install

#mkdir /var/log/clamav

#chown clamav:clamav /var/log/clamav

#vi /usr/local/etc/clamd.conf

```
LogFile /var/log/clamav/clamav.log
LogVerbose yes
LogTime yes
PidFile /var/run/clamd.pid
DatabaseDirectory /usr/local/share/clamav
```

#vi /usr/local/etc/freshclam.conf

```
DatabaseDirectory /usr/local/share/clamav
UpdateLogFile /var/log/clamav/freshclam.log
LogSyslog yes
LogVerbose yes
```

接著更新病毒碼至最新版本：

```
#/usr/local/bin/freshclam
```

接下來安裝 HAVP[25]：

```
#tar zxvf havp-0.92a.tar.gz
#cd havp-0.92a
#./configure
#make && make install
#groupadd havp && useradd -g havp -M havp
#chown havp:havp /var/log/havp /var/run/havp
#vi /usr/local/etc/havp/havp.config
```

```
ENABLECLAMLIB true
CLAMDBDIR /usr/local/share/clamav
TEMPDIR /tmp
SCANIMAGES false # 不使用圖片掃瞄
TRANSPARENT true
```

使用 1G 硬碟空間建立虛擬磁碟，以利進行檔案掃描之用：

```
#dd if=/dev/zero of=/root/havp_tmp.img bs=1024K count=1 \
seek=1024
#mkfs.ext2 /root/havp_tmp.img
#mount -o loop,mand /root/havp_tmp.img /var/tmp/havp
#chwon havp:havp /var/tmp/havp
```

更新 share library 資料庫：

```
#vi /etc/ld.so.conf
```

/usr/local/lib

#ldconfig

啟動 HAVP：

#cp ./havp-0.92a/etc/init.d/havp /etc/init.d

#/etc/init.d/havp strat

HAVP 需利用 iptables 的重導向功能將網頁封包導至 HAVP 進行過濾，需執行以下指令：

#iptables -t nat -A PREROUTING -i br0 -p tcp --dport 80 –j \
REDIRECT --to-port 8080

可將上述虛擬磁碟掛載、havp 啟動指令稿及重導向指令寫入 /etc/rc.d/rc.local 中，開機即會自動執行。

4.4. 虛擬化網路安全縱深防禦架構功能測試

4.3 節建置完成後，我們將使用一些工具來測試虛擬化網路安全縱深防禦架構之效果。

防火牆系統功能測試

圖 27 為我們使用攻擊者電腦開啟 IE 連線網頁伺服器之畫面，圖中可以看到連線至 140.10.10.10 可以開啟正常網頁。

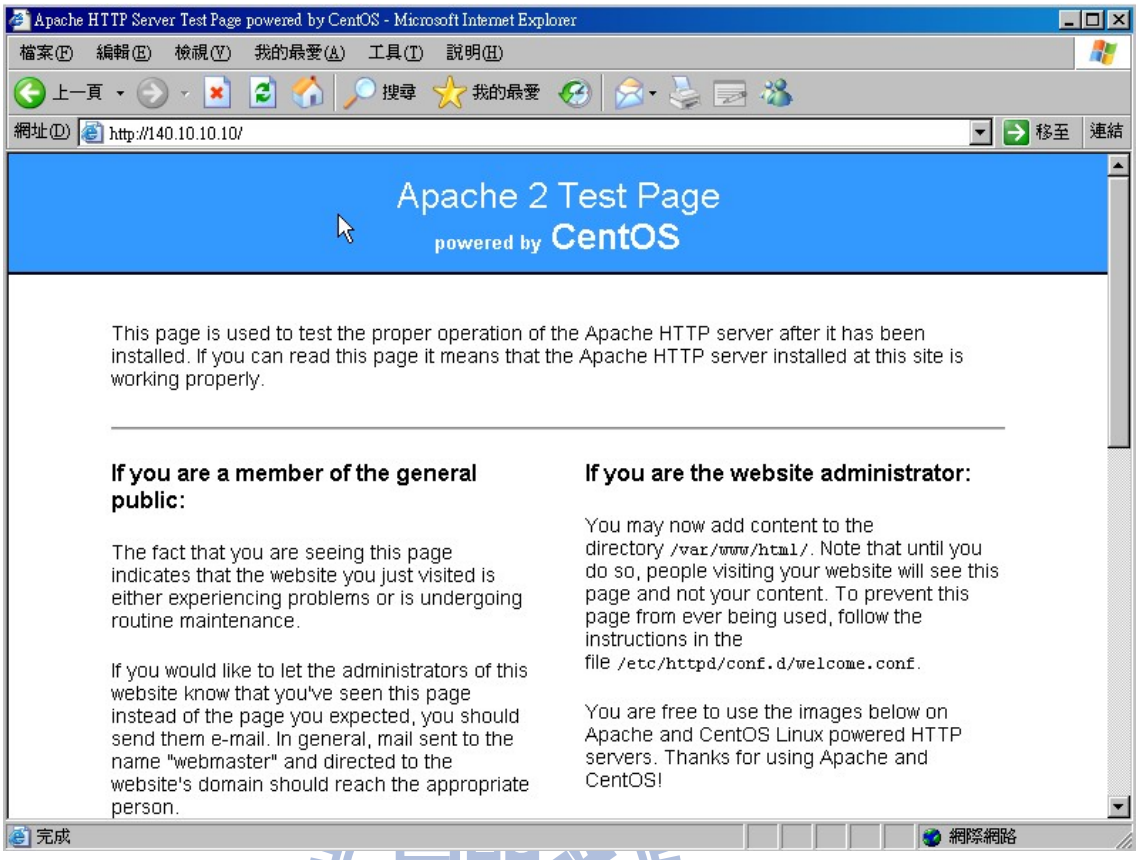

圖 27 測試網頁伺服器之連線

而圖 28 為遠端終端連線程式 Putty 的畫面，我們連線至網頁伺服器 140.10.10.10，並選擇以 Telnet 的方式進行連線。

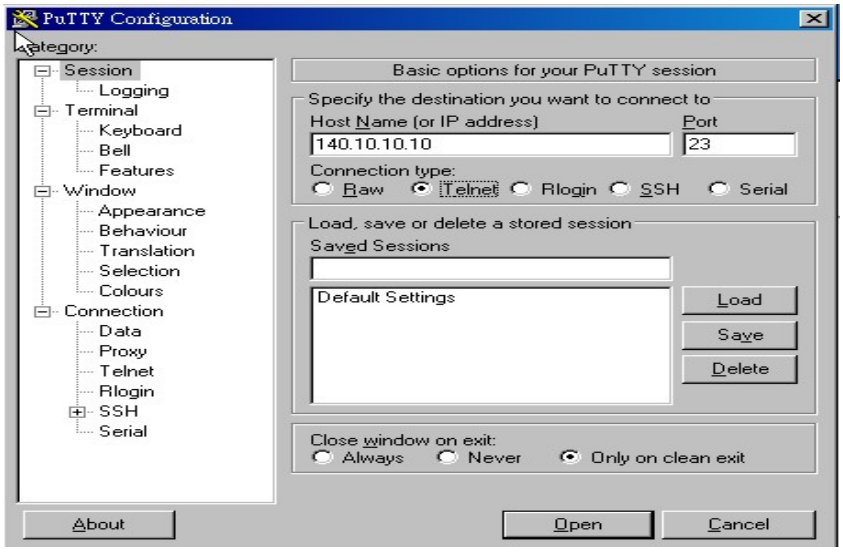

圖 28 利用 Putty 連線網頁伺服器的 Telnet 服務

當我們使用 Telnet 進行連線時會發現無法正常連線，因為防火牆有設定規則，只有開放 80 網路通訊埠可以連線，其餘連線封包一律丟棄，如圖 29 所示，我們可以看到封包之方向性和來源目的位址及網路通訊埠，IN=eth1,OUT=eth0，表示從外連進內部網路的封包，而 IN=eth0,OUT=eth1 表示封包是由內往外送的封包，SRC 表示來源ＩＰ位址，DST 表示目的 IP 位址，SPT 表示來源網路通訊埠，DPT 表示目的網路通訊埠。

```
[root@Firewall iptables]# vi iptables.rule
[root@Firewall iptables]# tail -f /var/log/messages
Feb 13 09:24:35 Firewall kernel: ACCEPT state packet: IN=eth0 OUT=eth1 SRC=192.168.0.10 DST=140.100.100.5 LEN=1500 TOS=0x00 P
REC=0x00 TTL=63 ID=33095 DF PROTO=TCP SPT=80 DPT=1172 WINDOW=6432 RES=0x00 ACK URGP=0
Feb 13 09:24:35 Firewall kernel: ACCEPT state packet: IN=eth1 OUT=eth0 SRC=140.100.100.5 DST=192.168.0.10 LEN=40 TOS=0x00 PRE
C=0x00 TTL=126 ID=1986 DF PROTO=TCP SPT=1172 DPT=80 WINDOW=64240 RES=0x00 ACK URGP=0
Feb 13 09:24:35 Firewall kernel: ACCEPT state packet: IN=eth0 OUT=eth1 SRC=192.168.0.10 DST=140.100.100.5 LEN=1500 TOS=0x00 P
REC=0x00 TTL=63 ID=33096 DF PROTO=TCP SPT=80 DPT=1172 WINDOW=6432 RES=0x00 ACK URGP=0
Feb 13 09:24:35 Firewall kernel: ACCEPT state packet: IN=eth0 OUT=eth1 SRC=192.168.0.10 DST=140.100.100.5 LEN=900 TOS=0x00 PR
EC=0x00 TTL=63 ID=33097 DF PROTO=TCP SPT=80 DPT=1172 WINDOW=6432 RES=0x00 ACK PSH FIN URGP=0
Feb 13 09:24:35 Firewall kernel: ACCEPT state packet: IN=eth1 OUT=eth0 SRC=140.100.100.5 DST=192.168.0.10 LEN=40 TOS=0x00 PRE
C=0x00 TTL=126 ID=1988 DF PROTO=TCP SPT=1172 DPT=80 WINDOW=64240 RES=0x00 ACK URGP=0
Feb 13 09:24:35 Firewall kernel: ACCEPT state packet: IN=eth1 OUT=eth0 SRC=140.100.100.5 DST=192.168.0.10 LEN=40 TOS=0x00 PRE
C=0x00 TTL=126 ID=1989 DF PROTO=TCP SPT=1172 DPT=80 WINDOW=64240 RES=0x00 ACK FIN URGP=0
Feb 13 09:24:35 Firewall kernel: ACCEPT state packet: IN=eth0 OUT=eth1 SRC=192.168.0.10 DST=140.100.100.5 LEN=40 TOS=0x00 PRE
C=0x00 TTL=63 ID=33098 DF PROTO=TCP SPT=80 DPT=1172 WINDOW=6432 RES=0x00 ACK URGP=0
Feb 13 09:33:53 Firewall kernel: DROP SSH,TELNET packet: IN=eth1 OUT=eth0 SRC=140.100.100.5 DST=192.168.0.10 LEN=48 TOS=0x00
PREC=0x00 TTL=126 ID=2237 DF PROTO=TCP SPT=1175 DPT=23 WINDOW=64240 RES=0x00 SYN URGP=0
Feb 13 09:33:56 Firewall kernel: DROP SSH,TELNET packet: IN=eth1 OUT=eth0 SRC=140.100.100.5 DST=192.168.0.10 LEN=48 TOS=0x00
PREC=0x00 TTL=126 ID=2238 DF PROTO=TCP SPT=1175 DPT=23 WINDOW=64240 RES=0x00 SYN URGP=0
Feb 13 09:34:02 Firewall kernel: DROP SSH,TELNET packet: IN=eth1 OUT=eth0 SRC=140.100.100.5 DST=192.168.0.10 LEN=48 TOS=0x00
PREC=0x00 TTL=126 ID=2243 DF PROTO=TCP SPT=1175 DPT=23 WINDOW=64240 RES=0x00 SYN URGP=0
```

圖 29 Iptables 防火牆日誌

所以從圖 29 中可以看出，當 140.100.100.5 連到 140.10.10.10 會被換成 192.168.0.10，因此 4.2 節防火牆功能模組時有提到會先經過 PREROUTING Chain，因此會把 IP 先 NAT 成內部 IP 才進行 FORWARD Chain 的過濾，因此看到的 LOG 就變成了內部 IP，所以從 140.100.100.5 到 192.168.0.10 的 80 網路通訊埠是被防火牆規則所接受的，所以會有 ACCEPT 的字樣，而從 140.100.100.5 到 192.168.0.10 的 23 網路通訊埠因為不被防火牆規則所接受而被丟棄

了，因此會有 DROP 的字樣。

入侵防禦系統功能測試

本研究使用 N-Stalker[23]這套免費版本的網頁應用程式測試軟體對 Snort 進行觸發警報的動作，圖 30 為我們在攻擊者電腦上執行 N-Stalker 的畫面。

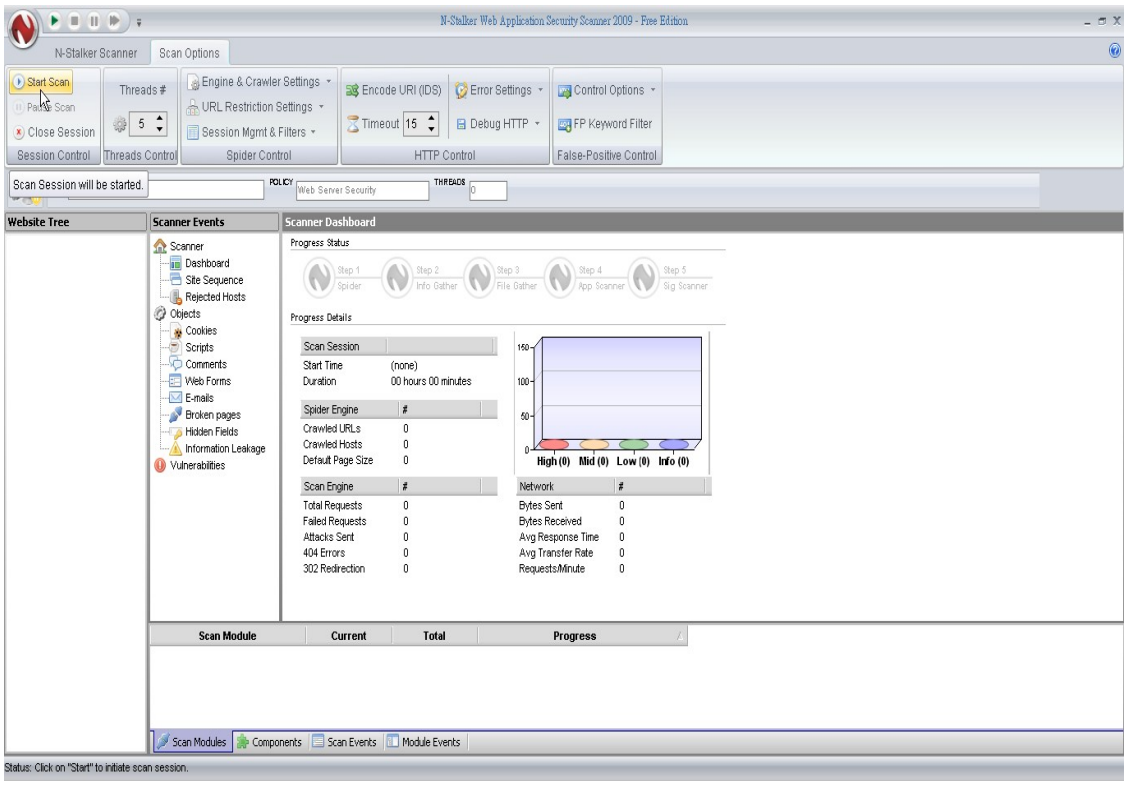

圖 30 N-Stalker 執行畫面

而我們可以從圖 31 看到由 N-Stalker 所觸發的攻擊事件，因為有偵測 N-Stalker 所產生的攻擊流量，因此可以證明封包有流向 Snort 並且進行阻擋之動作。

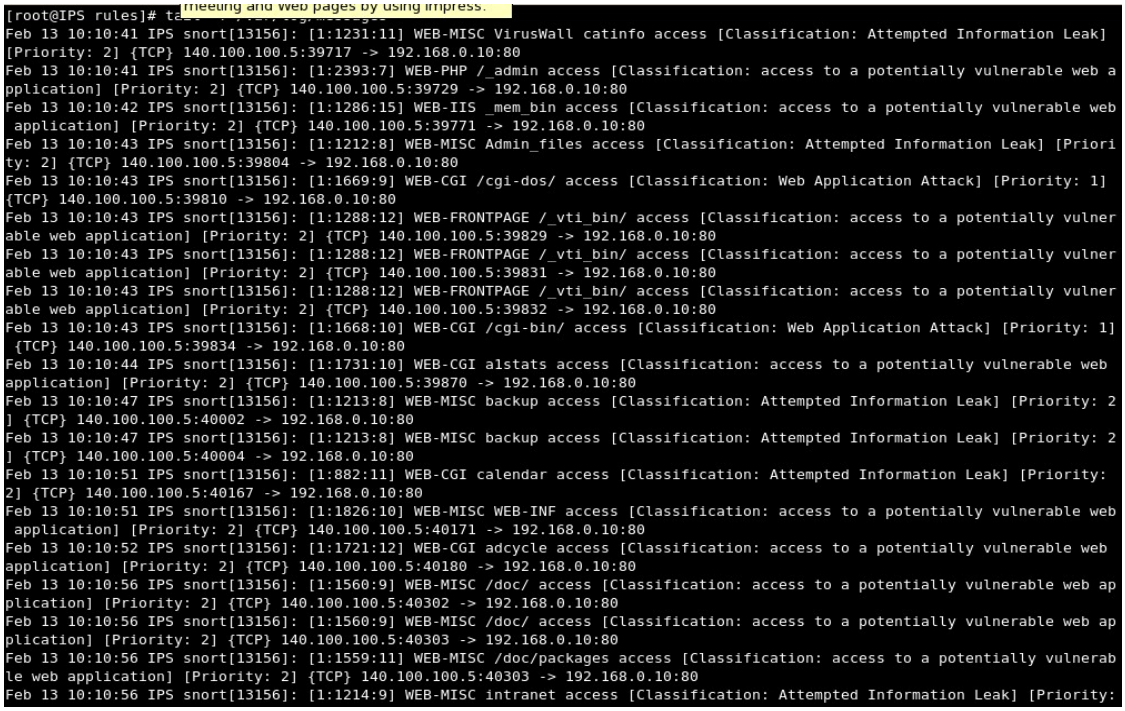

圖 31 Snort 入侵防禦系統日誌

防毒牆系統功能測試

圖 32 為我們在攻擊者電腦上做的一個惡意網頁，我們從 eicar[21] 網站上抓取病毒樣本檔放在攻擊者端的網頁伺服器中，然後利用內部網路的電腦上之瀏覽器連線到 140.100.100.5。

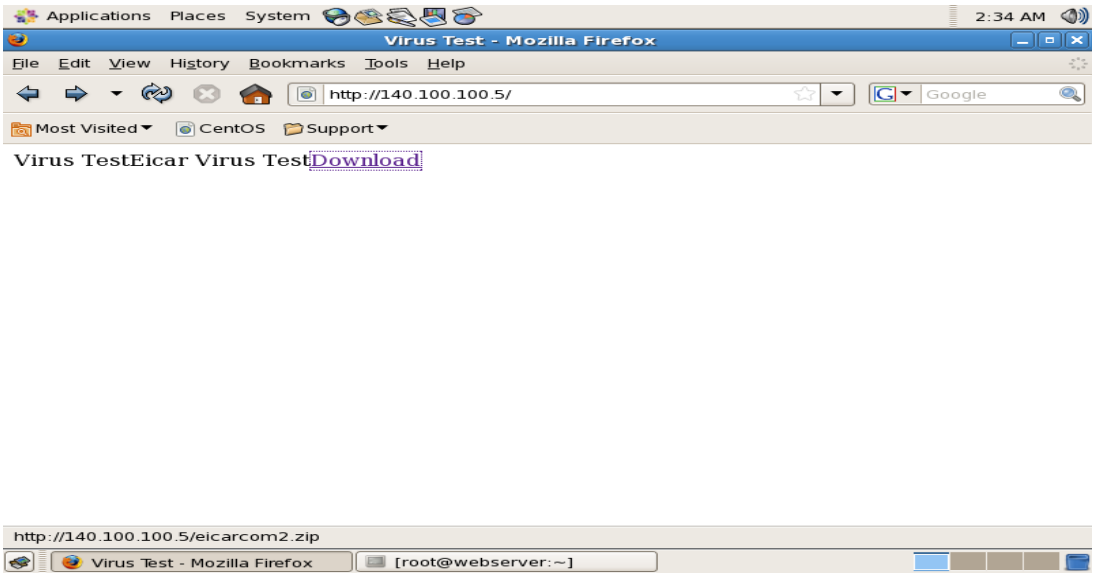

圖 32 攻擊者電腦之惡意程式網站

點選 Download 連結後，會進行 eicarcom2.zip 病毒樣本檔的下載，之後在瀏覽器上出現如圖 33 的畫面。

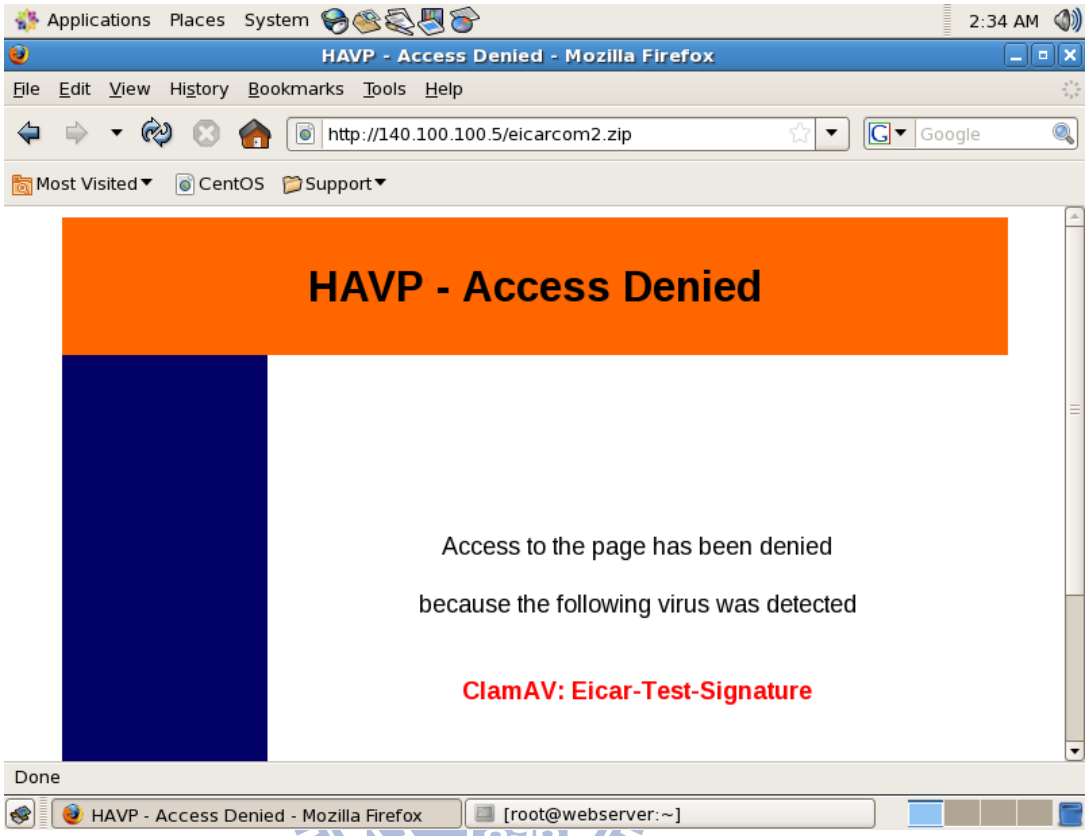

圖 33 HAVP 回應用戶病毒被阻擋之訊息頁面

由圖 33 所擷取下來的畫面可以證明 HAVP 防毒牆有正常運作，封包到達防毒牆時，會被重組回檔案，進行掃毒之後才決定是否放行該檔案進入內部網路。

進行上述三種測試後，可以證明利用虛擬化技術搭建的縱深防禦架構是可行的概念，但在效能上是否能夠擔起企業邊界安全防護的第一道關卡呢？本研究將在第五章進行各種網路安全防禦機制效能測試之分析及比較。

第五章　網路安全防禦機制之效能測試分析

第四章已說明了虛擬化網路安全縱深防禦架構是可以正常運行的一種防護架構，但為了了解其效能表現上是否也能如傳統實體網路安全縱深防禦機制和整合威脅管理系統一樣，本研究將於本章節進行相關的網路效能測試進行分析比較。

本章 5.1 節首先說明一下整個網路效能測試環境的實驗環境，並於 5.2 進行 Iperf 效能測試分析，接著在 5.3 節進行 Httperf 效能測試分析，最後於 5.4 節將本研究中所提到的網路安全防禦機制做一個總結比較。

5.1.　網路效能測試環境說明

圖 34 說明網路效能測試環境架構，利用前端之封包產生主機(PGHost)透過交換器連接網路安全防禦機制，然後交換器再將封包送到封包接收主機(Apache Web Server)。

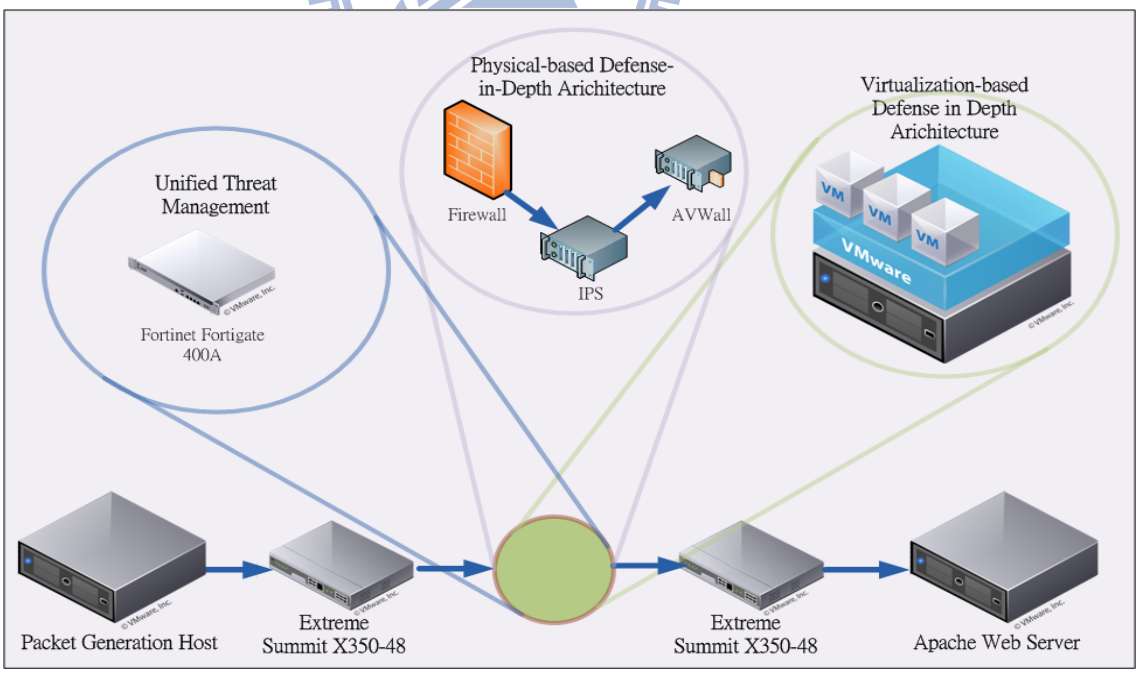

圖 34 網路效能測試環境示意圖

本研究所進行之網路安全防禦機制測試除了第四章的虛擬化網路

安全縱深防禦架構外，還須測試實體網路安全縱深防禦架構及整合威脅管理系統進行實測，而實驗環境的硬體環境如表 4 所示。

表 4 硬體設備與作業系統規格表

Firewall		IPS	
Model	IBM System X3250 M2	Model	IBM System X3250 M2
CPU	Intel xeon 2.66GHZ Quad Core	CPU	Intel xeon 2.66GHZ Quad Core
RAM	2GB DDR2 800MHZ	RAM	2GB DDR2 800MHZ
OS	CentOS 5.5	OS	CentOS 5.5
Kernel	Kernel-2.6.18.194.32.1	Kernel	Kernel-2.6.18.194.32
APP	Iptables-1.3.5	APP	Iptables-1.3.5 Snort -2.8.6
LAN	Intel PRO/1000 PT Dual	LAN	Intel PRO/1000 PT Dual
DISK	Segate 160G SAS x 2 (RAID0)	DISK	Segate 160G SAS x 2 (RAID0)
AntiVirus Wall		Web Server	
Model	IBM System X3250 M2	Model	IBM System X3250 M2
CPU	Intel xeon 2.66GHZ Quad Core	CPU	Intel xeon 2.66GHZ Quad Core
RAM	2GB DDR2 800MHZ	RAM	2GB DDR2 800MHZ
OS	CentOS 5.5	OS	CentOS 5.5
Kernel	Kernel-2.6.18.194.32.1	Kernel	Kernel-2.6.18.194.32.1
APP	Iptables-1.3.5 ClamAV-0.97 HAVP-0.92	APP	Apache-2.2.3-43 Iperf-2.0.4-1
LAN	Intel PRO/1000 PT Dual	LAN	Intel PRO/1000 PT Dual
DISK	Segate 160G SAS x 2 (RAID0)	DISK	Segate 160G SAS x 2 (RAID0)
Packet Generation Host		Extreme Switch	
Model	IBM System X3250 M2	Model	Summit X350-48t

CPU	Intel xeon 2.66GHZ Quad Core		
RAM	2GB DDR2 800MHZ	ExtremeXOS	12.3.1.2 v1231b2
OS	CentOS 5.5	PORT	48 GigabitEthernet ports
Kernel	Kernel-2.6.18.194.32.1		
APP	Iperf-2.0.4-1 Httperf-0.9.0-1	Fortinet UTM	
		Model	Fortigate 400A
LAN	Intel PRO/1000 PT Dual	IOS	3.0
DISK	Segate 160G SAS x 2 (RAID0)	PORT	6 GigabitEthernet ports

上述實驗環境架設完成後，要注意實體交換器上的配置，虛擬化網路安全縱深防禦架構在交換器上的配置如同 4.3 節說明，因此傳統網路安全縱深防禦架構必須要在實體交換器上設置 4 個 vlan 才能模擬出如 4.3 節所描述之封包傳送路徑進行網路效能測試，而整合威脅管理系統僅需設置 2 個 vlan 即可進行網路效能測試。

5.2. Iperf 效能測試分析

本研究使用 Iperf 網路效能測試工具測試網路防禦架構的實際吞吐量，並且利用 ping 指令輔助測試封包遺失率，最後利用 netstat 指令記錄所產生的封包數量。

實驗測試步驟

Iperf 測試方法需在 Web Server 主機上利用底下指令開啟測試服務：

```
#iperf –s –f a –l 1448k –w 128k –p 80
```

在封包產生主機(PGHost)利用底下指令進行網路效能測試：

```
#iperf –c server_ip –f a –t 20 –l {8,16,24,128,512,1448}k
–p 80 –P n
```

iperf 參數說明：

-c --client

-s --server

-f --format #bkmgaBKMGA

-l --len #The length of buffers to read or write, Iperf works by writing an array of len bytes a number of times. Default is 8 KB for TCP

-w --window

-p --port

-t --time #The time in seconds to transmit for. Default is 10 secs

-P --parallel # The number of simultaneous connections to make to the server

當封包產生主機(PGHost)開始送出 TCP 的流量時，同時需進行 ping 的測試作業，使用如下指令：

#ping –c 15 server_ip

也要同時利用底下指令記錄 Client 所送出的封包總數：

#netstat –IethX 1

在 iperf 程式跑完後，將所產生之總頻寬數據記錄下來，並且一併將 ping 指令產生之 packet loss 比率記錄下來，最後記錄 netstat 指令封包前後之數值，並代入底下公式產生每秒封包量之數值：

$$PPS = \frac{\left(Packets\_After - Packets\_Before\right)}{20}$$

Iperf 測試各網路防禦架構之分析說明

本研究將無防禦設備的情況做為網路效能測試的對照組，並在其間切換不同的網路防禦架構進行實驗測試，在虛擬化網路縱深防禦及傳統實體縱深防禦架構方面本研究各進行兩種版本的測試，一種為有 IPS 之版本，一種為無 IPS 之版本，因為在實驗過程中發現當有開啟 Snort-inline 模式時，網路效能會降低 8 成以上，其主要原因為 Snort 實作 inline 模式的原始程式碼有實作上的缺陷，因 Snort 原本是以入侵偵測之角度進行開發的，而 inline 模式是後期版本才附上去的功能，而且僅能使用一個佇列及一個執行緒進行檢測，因此不論實體設備規格如何，其資源使用上已被加以限制，因此才會對虛擬化縱深防禦及傳統實體縱深防禦架構再進行無 IPS 版本之網路效能測試，以證明其為 Snort-inline 模式的問題。

各種網路防禦架構之測試結果如附錄二所示。本研究將利用附錄二之實驗數據進行結果分析說明。底下以同時送 50 個連線數(P=50)為例進行吞吐量比較、每秒封包數比較及封包遺失率比較說明。

吞吐量(Throughput)之比較

從圖 35 可以看出各種網路防禦架構在不同封包大小時的吞吐量大致的狀況，無防禦設備之情形為最佳吞吐量，達 940Mbits/sec 左右，整合威脅管理系統也維持有 850Mbits/sec 至 880Mbits/sec，和無防禦設備相比算效能維持的相當不錯，而傳統縱深防禦(無 IPS)的情況平均維持在 810Mbits/sec 至 930Mbits/sec 之間的水準，而虛擬化縱深防禦(無 IPS)的部份平均約在 560Mbits/sec 至 600Mbits/sec 之間，以單台主機同時運行多種防禦系統來說，算是可以接受的範圍，而在傳統及虛擬化縱深防禦(有 IPS)的部份上述已討論過為何會形成這樣的結

果，因此不再贅述，而其平均之吞吐量僅維持在 50Mbits/sec 至 100Mbits/sec 之間，以中小企業對外線路僅 20Mbits/sec 以下的頻寬來說，還能夠負荷，但如果是大型企業的話就不敷使用了。

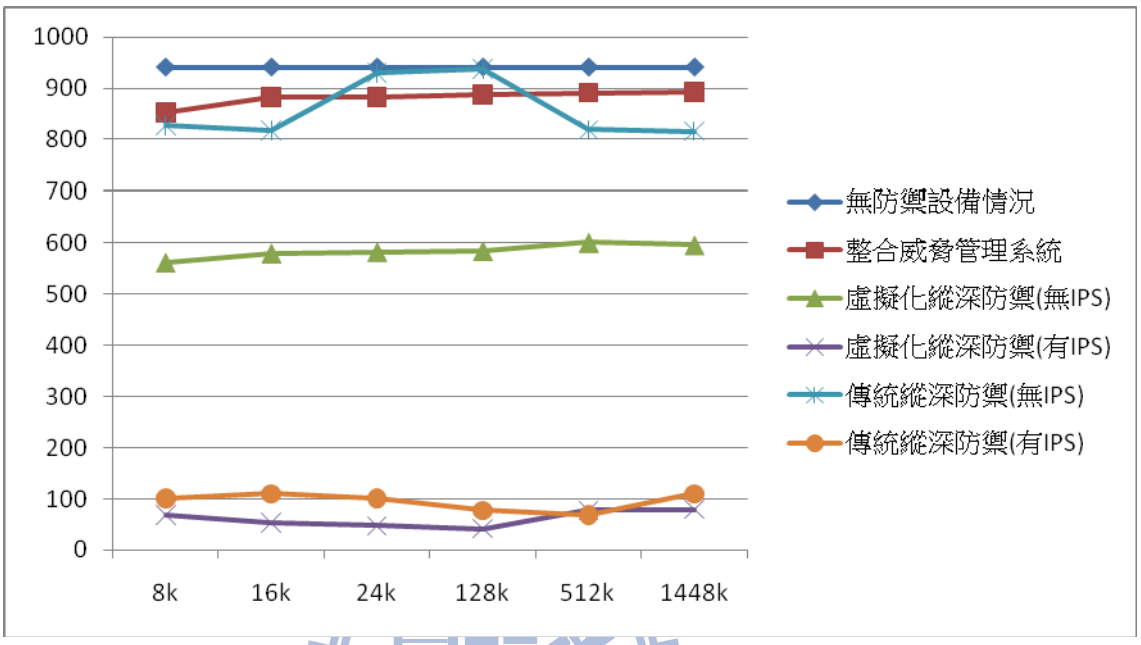

圖 35 網路效能吞吐量之比較圖

每秒封包數之比較

從圖 36 可以看出各種網路防禦架構在不同封包大小時的每秒封包數之情況，無防禦設備之情形為最大每秒封包數，每秒可送出 84000 個左右的封包，而在整合威脅管理系統的架構下，每秒封包數則維持在 77000 個至 78000 個左右，和無防禦設備相比算效能維持的相當不錯，而傳統縱深防禦(無 IPS)的情況大致和無防禦設備的情形差不多，平均每秒封包數維持在 84000 個左右，而虛擬化縱深防禦(無 IPS)的部份每秒封包數平均約在 52000 個至 57000 個之間，以單台主機同時運行多種開放式架構的防禦系統來說，算是有不錯的效能，而在傳統及虛擬化縱深防禦(有 IPS)的部份其平均每秒封包數僅介於 8000 個至 12000 個之間，且虛擬化縱深防禦架構略低於傳統縱深防禦架構。

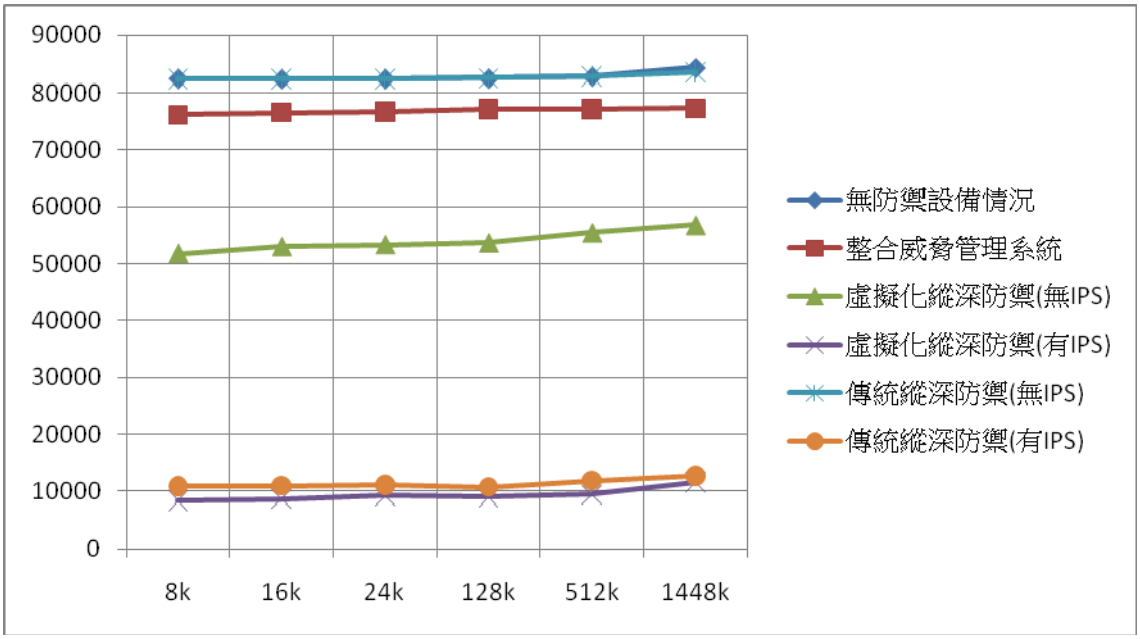

圖 36 網路效能每秒封包數之比較圖

封包遺失率(Packet Loss)之比較

從圖 37 可以看出各種網路防禦架構在不同封包大小時封包遺失率大致的情形，無防禦設備、整合威脅管理系統及傳統縱深防禦(無 IPS)之網路安全防禦架構皆沒有遺失任何封包，虛擬化縱深防禦(無 IPS)的部份僅在封包大小為 512K 時有遺失少量封包的情況，而在傳統及虛擬化縱深防禦(有 IPS)的部份其封包遺失比率較高，其為 IPS 只有在一個佇列中進行封包過濾之動作，且封包會依序進出佇列，所以造成封包傳送會有 Waiting Time，容易發生逾時現象，使封包遺失率提升。

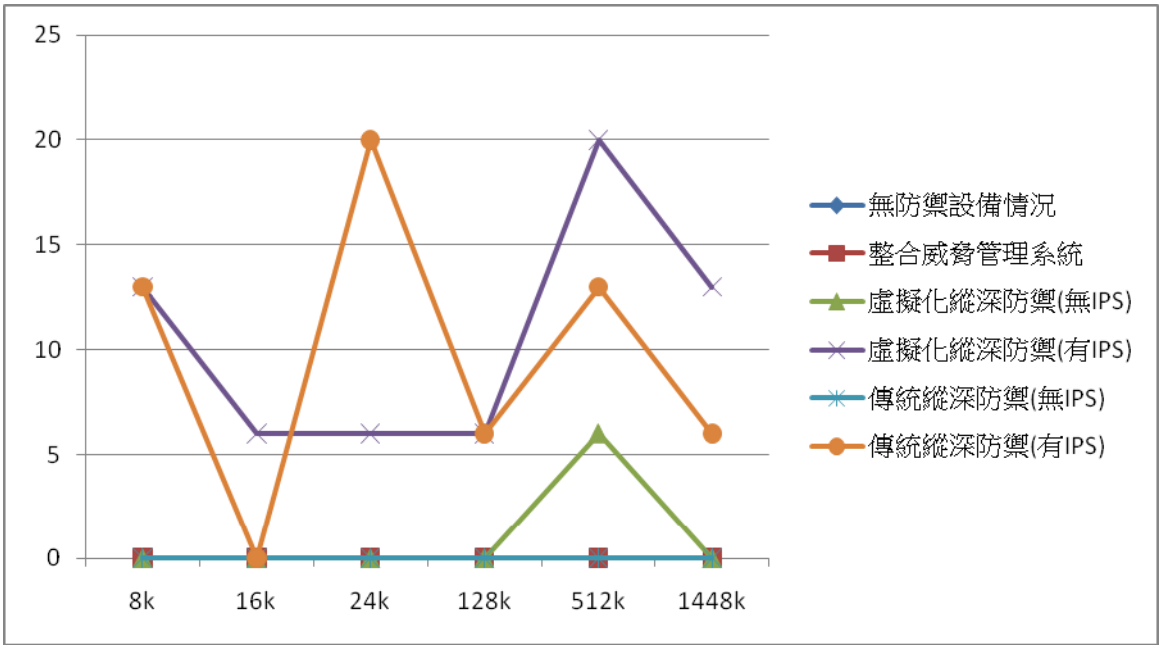

圖 37 網路效能封包遺失率之比較圖

5.3. Httperf 效能測試分析

本研究使用 Httperf 效能測試工具測試各種網路防禦架構的回應時間、每秒傳送之封包數以及 Request 封包回應之錯誤率，並於 Web Server 中放置兩個大小不同的檔案進行測試(15k 及 45k)，以利網路效能測試使用。

實驗測試步驟

Httperf 測試方法需在 Web Server 主機上開啟 Httpd 服務，指令如下：

```
#/etc/init.d/httpd start
```

利用 dd 指令產生 15k 及 45k 之檔案，並將上述二個檔案放置於 /var/www/html 目錄底下，指令如下：

```
#dd if=/dev/zero of=/var/www/html/15k bs=1k count=15
#dd if=/dev/zero of=/var/www/html/45k bs=1k count=45
```

dd 參數說明：

if --input

of --output

bs --bytes

count --bloc

在封包產生主機(PGHost)上，撰寫一支 shell script 進行 Httperf 之網路效能測試，內容如下：

#vi runhttperf.sh

```
#/bin/bash
httperf --timeout=10 --client=0/1 --server=$1 --port=80
--uri=/$2 --rate=$3 --send-buffer=4096 --recv-buffer=16384
--num-conns=$4 --num-calls=$5 > httperf.$2.$3.$4.$5.txt
```

httperf 參數說明：

--server : Web Server IP

--uri : File Location

--rate : Connection per Second

--num-conn : Total Connection

--num-calls : Request per Connection

以每秒 500 個 request 設定抓取 15k 檔案為例，可執行如下指令：

#./runhttperf.sh server_ip 15k 50 5000 10

測試完成後會得到如下的內容：

```
Total: connections 5000 requests 50000 replies 50000 test-duration 100.006 s

Connection rate: 50.0 conn/s (20.0 ms/conn, <=26 concurrent connections)
Connection time [ms]: min 18.5 avg 32.5 max 3039.8 median 26.5 stddev 90.8
Connection time [ms]: connect 4.1
Connection length [replies/conn]: 10.000
```

```
Request rate: 500.0 req/s (2.0 ms/req)
Request size [B]: 68.0

Reply rate [replies/s]: min 487.2 avg 500.0 max 515.6 stddev 6.2 (20 samples)
Reply time [ms]: response 1.4 transfer 1.4
Reply size [B]: header 250.0 content 15360.0 footer 0.0 (total 15610.0)
Reply status: 1xx=0 2xx=50000 3xx=0 4xx=0 5xx=0

CPU time [s]: user 21.36 system 78.62 (user 21.4% system 78.6% total 100.0%)
Net I/O: 7654.8 KB/s (62.7*10^6 bps)

Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

可從上述檔案得知 Response Time，並且能夠知道有多少個已送出的 request 封包但沒有收到 replies 封包之情況。每一次測試皆利用指令 netstat 將網路卡 RX 及 TX 的封包數及位元數記錄下來，指令如下所示：

#netstat –eI ethX 10

最後透過底下公式進行計算，得到 PPS：

$$PPS = \frac{(Packets\_After - Packets\_Before)}{20}$$

Httperf 測試各網路防禦架構之分析說明

本研究同 5.2 節一樣測試 4 種架構(6 種情況)，測試 6 種情況之理由也同 5.2 節一樣，是因為 Snort inline 模式效能不佳之問題，而各種網路防禦架構之測試結果如附錄三所示。本研究將利用附錄三之實驗數據進行結果分析說明。底下將以 15K 及 45K 兩種測試檔之結果進行每秒封包數比較、 回應時間比較以及 Request 封包回應之錯誤率比較說明。

每秒封包數之比較

圖 38 為利用 httperf 測試網站伺服器上 15K 的檔案大小之每秒封包數比較圖，從圖中可看出其成長曲線皆為 45 度角的直線，表示各種網路防禦架構的每秒封包數在每秒不同數量的 request 封包測試下數據都非常的接近，其皆呈 45 度角的直線等比往上成長。表示在 15K 的檔案測試下，每秒封包數無太大的差異。

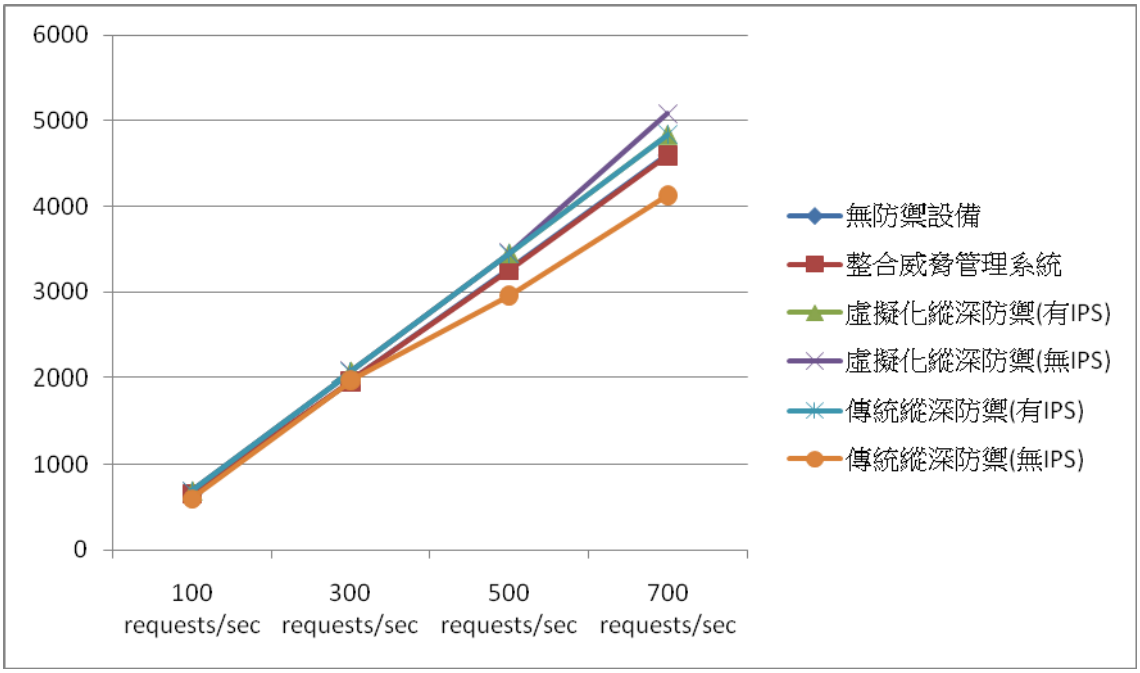

圖 38 網路效能每秒封包數之比較圖(15K)

圖 39 為利用 httperf 測試網站伺服器上 45K 的檔案大小之每秒封包數比較圖，從圖中可看出其成長曲線在每秒鐘 300 個 request 封包之後開始有了不一樣的變動，無防禦設備、整合威脅管理系統及傳統縱深防禦(無 IPS)大致上還是呈現 45 度角往上等比成長，只是傳統縱深防禦(無 IPS)成長的幅度偏緩，而虛擬化縱深防禦(有 IPS)在 4000 個封包左右即呈現平行，傳統縱深防禦(有 IPS)則在 6000 個封包數後也無法再往上成長，而虛擬化縱深防禦(無 IPS)在每秒鐘 500 個 request 封包後開始呈現負成長，這表示其主機資源已使用過量，因此無法維

持平盤。從每秒 8000 個 request 封包降至每秒 6000 個 request 封包。

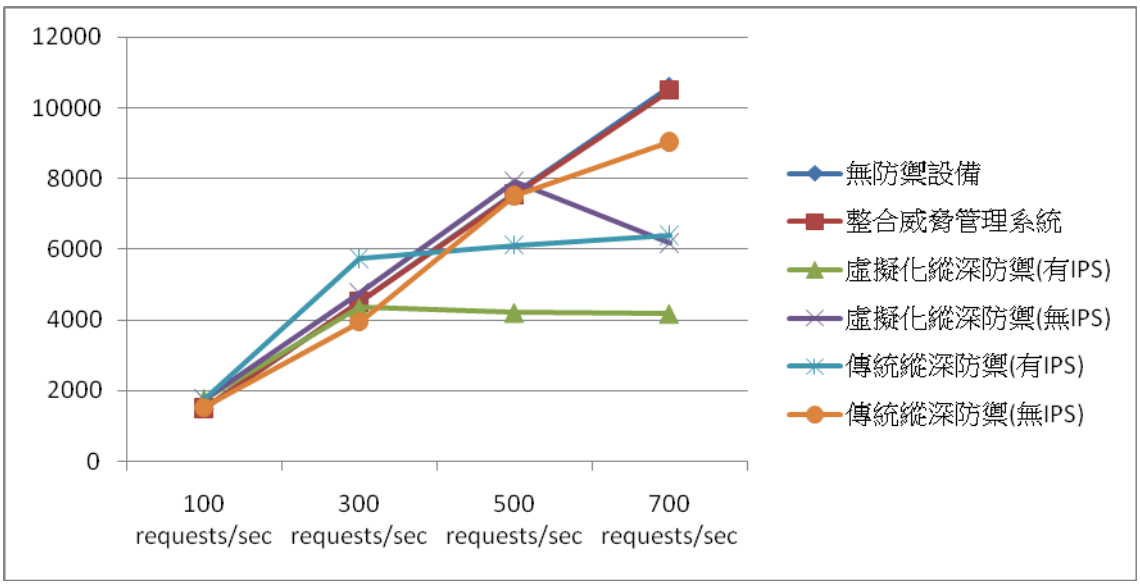

圖 39 網路效能每秒封包數之比較圖(45K)

封包回應時間之比較

圖 40 為利用 httperf 測試網站伺服器上 15K 的檔案大小之封包回應時間比較圖，從圖中左列可看出單位時間差異並不大，表示各種網路防禦架構的回應時間都很接近，在每秒 700 個 request 封包傳送下，以最快的無防禦設備之回應時間和最慢的虛擬化縱深防禦(有 IPS)之回應時間作比較，其差距頂多在 3ms 之內，所以網路效能封包回應時間在 15K 的檔案大小時，回應時間的差異性並不大。

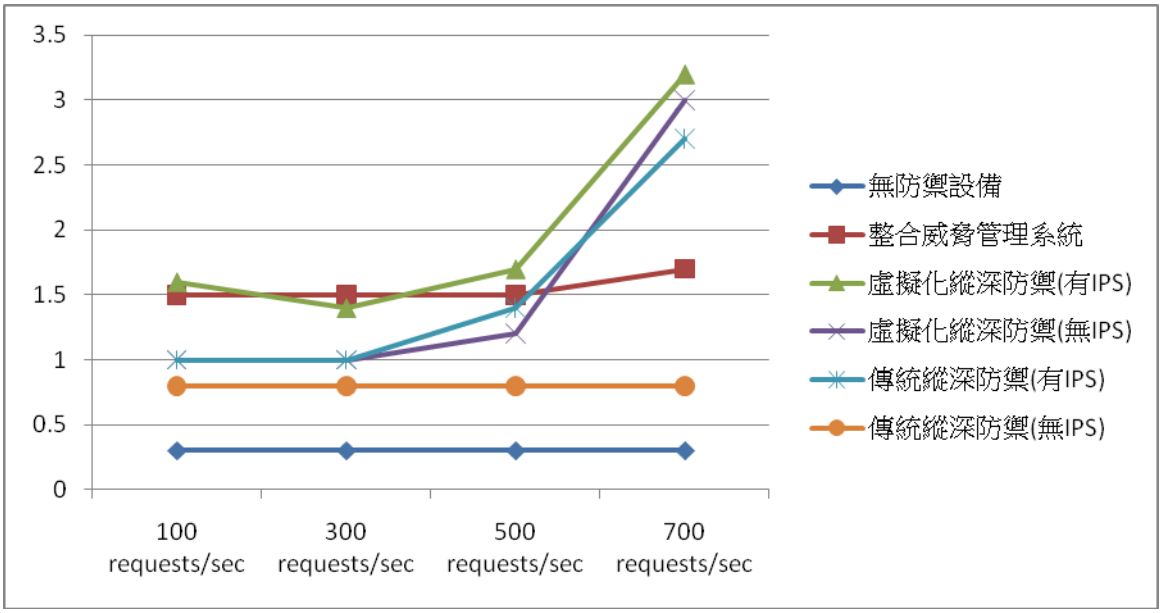

圖 40 網路效能封包回應時間之比較圖(15K)

圖 41 為利用 httperf 測試網站伺服器上 45K 的檔案大小之封包回應時間比較圖，從圖中左列可看出單位時間差異量較大，表示各種網路防禦架構的回應時間在每秒送出不同 request 數的測試下，有非常大的差距，從圖中可清楚看到在無防禦設備及傳統縱深防禦(無 IPS)的架構下，回應時間平均為 1.8 左右，而在每秒 300 個 request 封包傳送下，在傳統縱深防禦(有 IPS)已上升至 200ms 以上，而虛擬化縱深防禦(有 IPS)更上升至 800ms 以上，而在每秒 500 個 request 封包傳送下，在傳統縱深防禦(有 IPS)已上升至 600ms 以上，而虛擬化縱深防禦(有 IPS)上升至 900ms 以上，在每秒 700 個 request 封包傳送下，在傳統縱深防禦(有 IPS)還是維持在 600ms 以上，而虛擬化縱深防禦(有 IPS)則已接近 1000ms，而整合威脅管理系統及虛擬化縱深防禦(無 IPS)的反應時間則往上升接近 500ms 左右，因此當檔案變大至 45K 時，其回應時間的差異性較大。

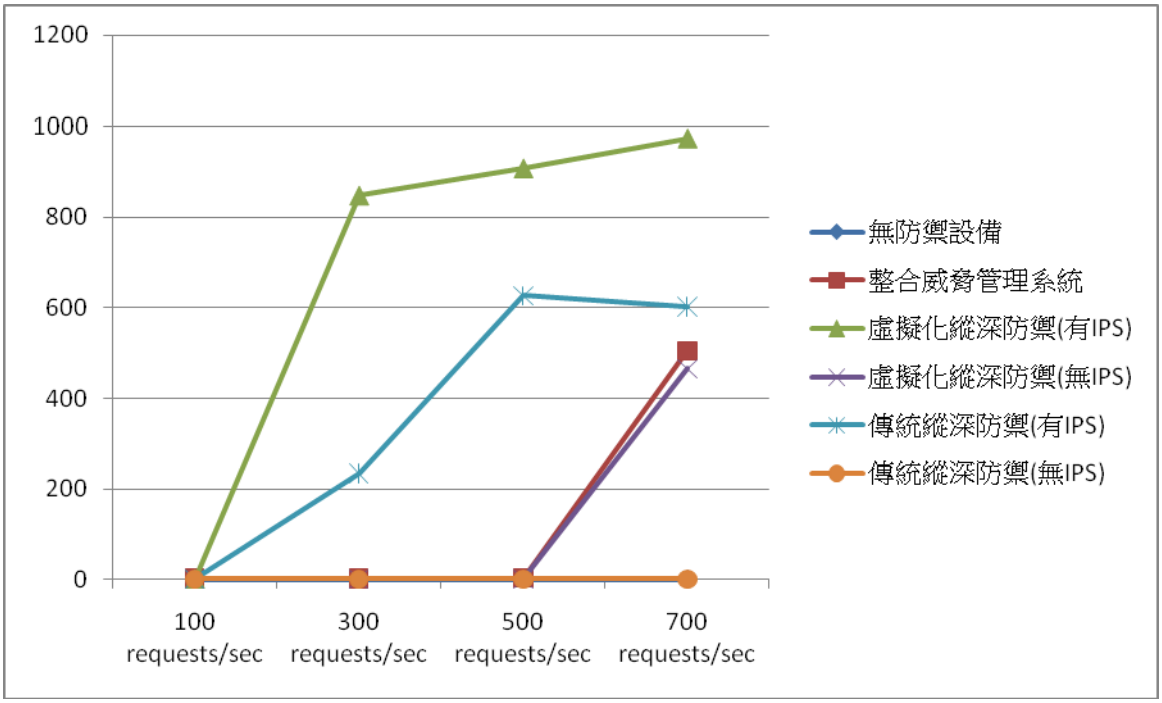

圖 41 網路效能封包回應時間之比較圖(45K)

Request 封包回應之錯誤率比較

圖 42 為利用 httperf 測試網站伺服器上 15K 的檔案大小之封包回應之錯誤率比較圖，從圖中可看出所有網路安全防禦架構都沒有遺失任何回應(replies)封包，所以網路效能封包回應之錯誤率在每秒鐘送出不同 request 封包要求伺服器上 15K 的檔案大小時，其錯誤率為 0。

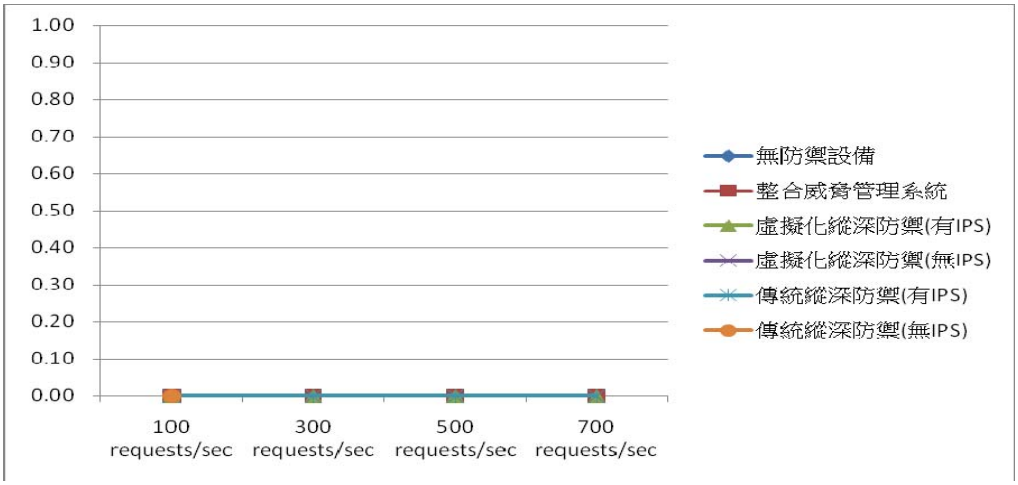

圖 42 網路效能 Request 封包回應之錯誤率比較圖(15K)

圖 43 為利用 httperf 測試網站伺服器上 45K 的檔案大小之封包回應之錯誤率比較圖，從圖中可清楚看到在每秒 100 個 request 封包傳送下，無防禦設備及傳統縱深防禦(無 IPS)的架構之錯誤率為 0，而在每秒 300 個 request 封包傳送下，在傳統縱深防禦(有 IPS)之錯誤率上升至 1%，而虛擬化縱深防禦(有 IPS)之錯誤率已上升至 5%，而在每秒 500 個 request 封包傳送下，在傳統縱深防禦(有 IPS)之錯誤率已上升至 7%，而虛擬化縱深防禦(有 IPS)則上升至 15%，在每秒 700 個 request 封包傳送下，在傳統縱深防禦(有 IPS)之錯誤率已上升至 13%，而虛擬化縱深防禦(有 IPS)之錯誤率則已上升至 24%，而整合威脅管理系統之錯誤率上升至 2%及虛擬化縱深防禦(無 IPS)之錯誤率則上升至 14%，因此當檔案變大至 45K 時，其 Request 封包回應之錯誤率的變化較大。

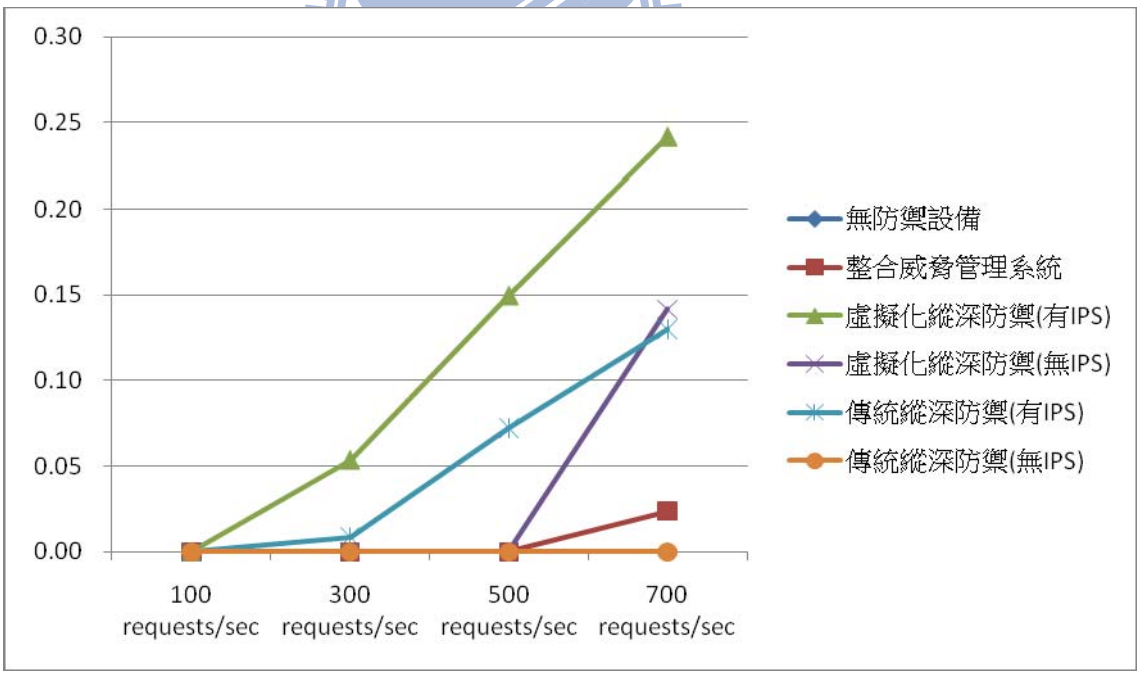

圖 43 網路效能 Request 封包回應之錯誤率比較圖(45K)

5.4. 網路安全防禦機制之比較

從 5.2 節及 5.3 節的網路效能測試分析下，可以發現整體的網路效能以整合威脅管理系統最好，傳統網路安全縱深防禦架構維持在中上水準，而虛擬化網路安全縱深防禦架構之表現也有中等水準，達可接受的標準，但這個比較的基準點稍微有些不平等的地方在於傳統網路安全縱深防禦架構使用三台工業型電腦，其 CPU 核心數量(每台四核心)較虛擬化縱深防禦架構只使用一台工業型電腦且每一虛擬機器僅使用一顆 vCPU 要來的多，而整合威脅管理系統其內部有非常多的 ASIC 在協助處理封包運算，其內建的防禦機制也都使用了硬體加速的方式提升效能，因此網路效能分析的結果僅為參考，並不能因為這樣的結果就評判虛擬化縱深防禦架構較整合威脅管理系統網路效能差。

以長遠硬體擴充性考量，虛擬化縱深防禦架構在擴充資源的部份是最佳的，其能在不影響縱深防禦架構運作時直接增加運算資源，例如：如果使用刀鋒伺服器建置虛擬化縱深防禦架構，則可動態調整 CPU 數量、記憶體大小及網路卡資源給虛擬機器使用，且不會對防禦架構造成任何影響，當一台不敷使用時，只要把虛擬機器進行複製並加入運作即可，大幅提升企業資源可用率及可用性，傳統縱深防禦架構要增加資源只能增加有限的記憶體或更換 CPU 以提升其速度，但對整體的效能提升通常不太顯著，因為能升級的部份是有限的，如果是以實體擴充的方式一直增加硬體資源，企業的營運成本將大幅提升，而威脅整合管理系統無法進行資源的升級，且企業網路架構成長到一定程度時，就必須進行汰換，更不符合經濟效益。

從電力資源的角度考量，從本次虛擬化縱深防禦架構來看，一台運行 VMWare ESX Server 的 IBM System X3250 M2 其電源供應器耗電

約為 350 瓦。傳統縱深防禦架構共 3 台 IBM System X3250 M2，因此用電量約為 1050 瓦，整合威脅管理系統僅使用 50 瓦的電力，因其不含硬碟及其他週邊設備的關係，若再加上冷氣、空間、人力維護、備份復原時間節省之成本，每年將可節省下不少費用。採用虛擬化縱深防禦的方式，可以有效降低主機空間、冷氣、電力的龐大需求，也可配合節能省碳的政策方針，達到令人滿意的成效。

從高可用性的角度考量，整合威脅管理系統必須要購買兩台以上的硬體設備進行高可用性的部署，而傳統縱深防禦架構則是每一種防禦機制都要兩台以上，非常不符經濟效益，且部署也較為困難，實體的接線也比較複雜，而虛擬化縱深防禦架構則完全不需要購買任何設備，只要底層硬體資源足夠，要達到高可用性較其他防禦架構簡單，且不需複雜的接線，部署非常簡易。

從硬體成本投資的角度考量，由我們的實驗環境可知，利用一台 IBM System X3250 M2 工業型電腦即可建置虛擬化縱深防禦架構，傳統縱深防禦架構就需要三台，成本就多三倍，整合威脅管理系統則因屬特製的硬體設備，因此其價格更是昂貴，大約是單台工業型電腦的十倍價錢左右，因此如果以購買整合威脅管理系統的成本建置虛擬化縱深防禦架構，將可獲得更多的電腦資源。

而從資源使用量來看，整合威脅管理系統及虛擬化網路安全縱深防禦架構皆能有效運用本機資源，而虛擬化平台還能夠對不同防禦設備依其資源使用量動態調整，因此較整合威脅管理系統有彈性，而傳統網路安全縱深防禦架構之防禦設備本機資源使用量較低，常會有浪費資源之問題存在。

表 5 網路安全防禦機制之比較

	虛擬化網路安全縱深防禦架構	實體網路安全縱深防禦架構	整合威脅管理系統
縱深防禦能力	高	高	低
本機資源使用率	高	低	高
擴充性	高	中	低
高可用性	高	低	中
電力成本	低	高	低
網路效能	中	中上	高

表 5 是本研究根據上述比較分析後的結果所呈現的一張比較表，其簡單說明各種防禦機制在網路效能、電力用量、擴充性、高可用性、硬體成本及資源使用考量下之優劣，可做為企業在評估建構網路安全防禦機制時的參考。而從 3.2 節相關研究中，我們可以發現類似的虛擬化技術已經蓬勃發展，所提供的功能也愈來愈強大豐富，過去虛擬化技術效能不彰的問題將成為歷史。可以預見的是伺服器的效能將不斷的提昇，會有愈來多空閒的資源閒置不能妥善利用，而不斷的購買伺服器代表不斷地增加伺服器的持有成本(空間、電力、空調)，此時導入虛擬化技術將可協助企業改善上述問題。

第六章　結論 (Conclusion)

第五章的效能測試比較，說明了本研究-以虛擬化技術為基礎之網路安全縱深防禦架構的優劣，而就對企業來說，損失一些網路效能可換取更多的優點是值得的，因此於 6.1 節說明本研究對企業資訊基礎建設有何效益，並且於 6.2 節說明本研究可再更精進的地方。

6.1. 對企業資訊基礎建設之效益說明

本研究結合了網路安全縱深防禦機制及虛擬化技術，除了能夠提升企業之網路安全外，還可以降低企業營運成本以及減少運算資源之浪費，利用縱深防禦的概念強化企業網路安全架構，因此可提高安全性，利用虛擬化技術，可有效節省電力成本、空間成本以及資源擴充成本。以往企業的軟硬體設施可能只能使用 3~4 年即要汰換，每次的汰換就是一次大工程，汰換下來的東西也將造成浪費，而虛擬化技術可有效整合這類型資源，因此可使企業投資成本降低，而虛擬化技術也可增加資源利用率，以前購買一台 1U 的伺服器可能只會用到其 15~20%左右的資源(不論是 CPU、記憶體及磁碟等)，但現在使用虛擬化技術，可根據伺服器資源使用量進行調整，使企業投資的金額皆花費在刀口上。機房機架機櫃空間也將大幅節省，連帶空調也就不需調到太低的溫度，因為實體伺服器的減少，使得電力使用量可大幅降低，符合現今綠化機房之需求。

另外本研究之虛擬化縱深防禦架構還有另一項優勢就是高可用性，其可做到零斷線甚至是系統容錯(system fault tolerance)等級的備援，且其後端使用高速的光纖儲存網路(SAN)也提升了企業資料的可靠性。

6.2. 未來工作

本研究尚有一些留待未來研究之問題，像是虛擬化縱深防禦架構沒有一個主控台，因此未來可以建議開發一套網管系統針對所有虛擬化防禦機制進行統一的控管，另外 Snort inline mode 的問題也值得我們後續進行探討，如果將其進行程式碼最佳化，助其效能提升相信也會是一個有趣之議題，例如將其改成同時多個執行緒能同時對多個佇列進行過濾分析，這樣效能應該會有顯著之提升。

參 考 文 獻

[1] D. Kewley, J. Lowry, “Observations on the effects of defense in depth on adversary behavior in cyber warfare”, Proceedings of the IEEE SMC Information Assurance Workshop, West Point, New York, June 2001.

[2] Neiger, G., A. Santoni, F. Leung, D. Rodgers, and R. Uhlig, Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. Intel Technology Journal, 10 August 2006. 10(03): p. 167-178.

[3] R. Hiremane. Intel Virtualization Technology for Directed I/O (Intel VT-d). Technology@Intel Magazine, 4(10), May 2007.

[4] Y. Koh, C. Pu, Y. Shinjo, H. Eiraku, G. Saito, D. Nobori, “Improving Virtualized Windows Network Performance by Delegating Network Processing”, In *Proceedings* of the IEEE Conference on Network Computing and Applications (NCA), 2009.

[5] S. Nanda and T. Chiueh, “A survey of virtualization technologies,” Stony Brook University, Tech. Rep. TR-179, Feb. 2005.

[6] C-H. Hsu and U. Kremer, “A framework for automatic construction of performance predication models”, In Proceedings of the 1st Workshop on Feedback-Directed Optimization, October 1998.

[7] D. Mosberger and T. Jin, “httperf: A tool for measuring Web server Performance”, In First Workshop on Internet Server Performance (WISP), HP Labs report HPL-98-61, June 1998.

[8] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, and Y. Chen, “GreenCloud: a new architecture for green data center”, In 6th international conference industry session on Autonomic computing and

communications, pages 29-38. ACM New York, NY, USA, 2009.

[9] Bhattacharya, S.P., Apte, V, “A Measurement Study of the Linux TCP/IP Stack Performance and Scalability on SMP systems”, In Proceedings of the 1st International Conference on COMmunication Systems softWAre and middlewaRE (COMSWARE), New Delhi (2006).

[10] G. Vallee, T. Naughton, C. Engelmann, H. Ong, and S. Scott, “System-level virtualization for high performance computing,” Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on, pp. 636–643, Feb. 2008.

[11] R. P. Lippmann et al, “Validating and restoring defense in depth using attack graphs”, In *Proceedings of MILCOM 2006*, Washington, DC.

[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization”, In SOSP ’03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164–177, New York, NY, USA, 2003. ACM.

[13] 黃能富，朱彥銘，林元偉，陳昶安，陳世興，賴昇鴻。網路安全縱深防禦系統之研製。行政院國家科學委員會專題研究計畫成果報告，已出版，新竹市。

[14] 曾宇瑞（2000）。網路安全縱深防護機制之研究。國立中央大學資訊管理研究所碩士論文，未出版，桃園縣。

[15] 吳金庭(2008)。以 Snort 偵測並封鎖網路異常行為之研究。國立交通大學理學院碩士論文。未出版，新竹市。

[16] 周海刚，邱正伦，肖军模(2005)。网络主动防御安全模型及体系结构。解放军理工大学通信工程学院論文，解放军理工大学学报（自

然科学版），江苏南京。

[17]RFC 1918- Address Allocation for Private Internets, "http://www.faqs.org/rfcs/rfc1918.html".

[18]Wikipedia- Firewll(computing), "http://en.wikipedia.org/wiki/Firewall_(computing)".

[19]Wikipedia- Virtualization, "http://en.wikipedia.org/wiki/Virtualization".

[20]Wikipedia- Xen, "http://en.wikipedia.org/wiki/Xen".

[21]Eicar, "http://www.eicar.org".

[22]CentOS, "http://www.centos.org/".

[23]N-Stalker The Web Security Specialists, "http://www.nstalker.com/".

[24]Snort, "http://www.snort.org/".

[25]HAVP - HTTP Anti Virus Proxy - The web antivirus solution, "http://www.server-side.de/".

[26]ClamAV, "http://www.clamav.net/lang/en/".

附　錄　一

底下為本次實驗中，有關防火牆 Iptables 之相關規則及設定。

Iptables.rule

```
#!/bin/bash
# Enviroment Varible
EXTIF="eth0"    # External Interface
INIF="eth1"    # Internal Interface
# Internal Network
EXTNET=”140.10.10.0/24”
INNET="192.168.0.0/24"
EXTWEB="140.10.10.10"
INWEB="192.168.0.10"
export EXTIF INIF EXTNET INNET EXTWEB INWEB
# For Local Firewall Setting
# 1.System parameter setting
echo "1" > /proc/sys/net/ipv4/tcp_syncookies
echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
                echo "1" > $i
done
for i in /proc/sys/net/ipv4/conf/*/log_martians; do
                echo "1" > $i
done
for i in /proc/sys/net/ipv4/conf/*/accept_source_route; do
                echo "0" > $i
done
```

```
for i in /proc/sys/net/ipv4/conf/*/accept_redirects; do
        echo "0" > $i
done
for i in /proc/sys/net/ipv4/conf/*/send_redirects; do
        echo "0" > $i
done
# 2.Clear rule,default policies,loopback interface allow
PATH=/sbin:/usr/sbin:/bin:usr/bin; export PATH
iptables -F
iptables -X
iptables -Z
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state RELATED -j ACCEPT
# 3.Start extra firewall script moudles
if [ -f /usr/local/iptables/iptables.deny ]; then
        sh /usr/local/iptables/iptables.deny
fi
if [ -f /usr/local/iptables/iptables.allow ]; then
            sh /usr/local/iptables/iptables.allow
fi
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
# 4.Allow some kinds of ICMP packets incomming
AICMP="0 3 3/4 4 11 12 14 16 18"
for tyicmp in $AICMP
```

```
do
iptables -A INPUT -i $EXTIF -p icmp --icmp-type $tyicmp -j ACCEPT
done
# 5.Allow some kinds of service incomming
iptables -A INPUT -p TCP -i EXTIF --dport 22 -j ACCEPT
# For Internal Server firewall rules setting
# 1.Load moudles
modules="Ip_tables ip_nat ip_nat_ftp ip_nat_irc ip_conntrack \
  ip_conntrack_ftp ip_conntrack_irc"
for mod in $modules
do
                testmod=`lsmod |grep "{$mod}"`
                if [ "$testmod" == "" ]; then
                modprobe $mod
                fi
done
# 2.Clear NAT Table rule
iptables -F -t nat
iptables -X -t nat
iptables -Z -t nat
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT
# 3.Open ip_forward
if [ "$INIF" != "" ]; then
                iptables -A INPUT -i $INIF -j ACCEPT
                echo "1" > /proc/sys/net/ipv4/ip_forward
```

```
if [ "$INNET" != "" ]; then
                    for innet in $INNET
                    do
                        iptables -t nat -A POSTROUTING -s $innet -o \
                        $EXTIF -j MASQUERADE
                    done
                fi
fi
# 4.Internal Server static NAT
iptables -t nat -A PREROUTING -i $EXTIF -d $EXTWEB -j \
    DNAT --to $INWEB
iptables -t nat -A POSTROUTING -o $EXTIF -s $INWEB -j \
    SNAT --to $EXTWEB
# 5.Proxy-ARP
arp -i $EXTIF -s $EXTWEB xx:xx:xx:xx:xx:xx pub
route add -host $EXTWEB $INIF
```

Iptables.allow

```
#!/bin/bash
# Local Chain
iptables -A INPUT -i $EXTIF -s $EXTNET -j ACCEPT
iptables -A INPUT -i $INIF -s $INNET -j ACCEPT
# Forward Chain
# Out to In
iptables -A FORWARD -i $EXTIF -o $INIF -m state --state \
ESTABLISHED,RELATED -j LOG --log-prefix "ACCEPT state \
packet: "
iptables -A FORWARD -i $EXTIF -o $INIF -m state --state \
ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i $EXTIF -o $INIF -p tcp -d $INWEB \
    --dport 80 -j LOG --log-prefix "ACCEPT web server packet: "
iptables -A FORWARD -i $EXTIF -o $INIF -p tcp -d $INWEB \
    --dport 80 -j ACCEPT
iptables -A FORWARD -i $EXTIF -o $INIF -p icmp -d $INWEB –j \
LOG --log-prefix "ACCEPT ICMP packet: "
iptables -A FORWARD -i $EXTIF -o $INIF -p icmp -d $INWEB –j \
ACCEPT
# In to Out
iptables -A FORWARD -i $INIF -o $EXTIF -p tcp --dport 80 –j LOG \
--log-prefix "ACCEPT go out web packet: "
iptables -A FORWARD -i $INIF -o $EXTIF -p tcp --dport 80 –j \
ACCEPT
iptables -A FORWARD -i $INIF -o $EXTIF -p icmp -j LOG \
--log-prefix "ACCEPT go out icmp packet: "
```

```
iptables -A FORWARD -i $INIF -o $EXTIF -p icmp -j ACCEPT
iptables -A FORWARD -i $INIF -o $EXTIF -m state –state \
ESTABLISHED,RELATED -j LOG --log-prefix "ACCEPT state \
packet: "
iptables -A FORWARD -i $INIF -o $EXTIF -m state –state \
ESTABLISHED,RELATED -j ACCEPT
```

Iptables.deny

```
#!/bin/bash
# Forward Chain
# Out to In
iptables -A FORWARD -i $EXTIF -o $INIF -m state –state INVALID \
-j LOG --log-prefix "DROP invalid packet: "
iptables -A FORWARD -i $EXTIF -o $INIF -m state –state INVALID \
-j DROP
# In to Out
iptables -A FORWARD -i $INIF -o $EXTIF -m state –state INVALID \
-j LOG --log-prefix "DROP invalid packet: "
iptables -A FORWARD -i $INIF -o $EXTIF -m state –state INVALID \
-j DROP
```

附　錄　二

附錄二為 iperf 根據本研究要求的 4 種架構(6 種情況)所測試出來的數據，以表 6 為例，此為無防禦設備之情況，只有 PGHost 及 Web Server 兩台主機直接測試的數據，我們可以看到在只有 1 個連線數到同時 100 個連線數的網路效能測試當中，不論封包大小多少，其吞吐量(Throughput)差異並不大，約 940Mbits/sec 左右，PPS(Packet per Second)約達 81000 個以上，而封包遺失率是 0%。其餘情況如表 7 至表 11 所示。

表 6 iperf 網路效能測試數據-無防禦設備

P=1	Payload size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	941	81302	0
	16k	941	81298	0
	24k	941	81300	0
	128k	941	81305	0
	512k	941	81305	0
	1448k	941	81358	0
P=5	Payload size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	941	81391	0
	16k	941	81403	0
	24k	941	81403	0
	128k	941	81397	0
	512k	941	81451	0
	1448k	941	81563	0
P=10	Payload size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	941	81515	0
	16k	941	81506	0
	24k	941	81512	0
	128k	941	81516	0
	512k	941	81633	0
	1448k	941	81923	0
P=20	Payload size	Throughput(Mbits/sec)	PPS	packet loss (%)

	8k	941	81705	0
	16k	941	81746	0
	24k	941	81733	0
	128k	941	81776	0
	512k	941	81889	0
	1448k	941	82027	0
P=50	Payload size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	941	82475	0
	16k	941	82403	0
	24k	941	82450	0
	128k	941	82489	0
	512k	941	82838	0
	1448k	941	84286	0
P=100	Payload size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	941	83578	0
	16k	941	83564	0
	24k	941	83521	0
	128k	941	83777	0
	512k	940	84514	0
	1448k	940	86960	0

表 7 iperf 網路效能測試數據-整合威脅管理系統

P=1	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	862.00	74982	0
	16k	863.00	75026	0
	24k	858.00	75039	0
	128k	866.00	74891	0
	512k	863.00	74903	0
	1448k	864.00	74976	0
P=5	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	865.00	75029	0
	16k	866.00	75186	0
	24k	867.00	75182	0
	128k	864.00	74963	0
	512k	865.00	74986	0

	1448k	865.00	75026	0
P=10	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	888.00	77028	0
	16k	875.00	76016	0
	24k	887.00	77405	0
	128k	895.00	77642	0
	512k	896.00	77701	0
	1448k	896.00	77647	0
P=20	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	882.00	76748	0
	16k	885.00	77016	0
	24k	888.00	77209	0
	128k	892.00	77482	0
	512k	894.00	77427	0
	1448k	893.00	77477	0
P=50	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	852.00	76172	0
	16k	882.00	76584	0
	24k	883.00	76702	0
	128k	888.00	77202	0
	512k	891.00	77227	0
	1448k	892.00	77327	0
P=100	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	869.00	75927	0
	16k	869.00	76323	0
	24k	872.00	76564	0
	128k	874.00	76868	0
	512k	878.00	77049	0
	1448k	883.00	77142	0

表 8 iperf 網路效能測試數據-虛擬化縱深防禦(無 IPS)

P=1	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	564	48,732	0
	16k	576	49,700	0
	24k	570	49,201	0

	128k	567	48,931	0
	512k	569	49,099	0
	1448k	588	50,740	0
P=5	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	580	50,467	0
	16k	612	53,159	0
	24k	600	52,146	0
	128k	589	51,204	0
	512k	629	54,746	0
	1448k	602	52,508	0
P=10	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	634	55,531	0
	16k	643	56,302	0
	24k	580	51,382	0
	128k	603	53,075	0
	512k	623	54,816	0
	1448k	610	53,846	0
P=20	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	583	51,896	13
	16k	600	53,915	6
	24k	610	54,515	0
	128k	604	54,192	6
	512k	607	54,488	0
	1448k	615	55,312	0
P=50	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	562	51,802	0
	16k	580	53,024	0
	24k	583	53,354	0
	128k	585	53,666	0
	512k	601	55,459	6
	1448k	596	56,788	0
P=100	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	591	55,486	6
	16k	585	54,945	0
	24k	603	56,491	0
	128k	595	56,105	6

	512k	576	55,151	20
	1448k	606	59,399	0

表 9 iperf 網路效能測試數據-虛擬化縱深防禦(有 IPS)

P=1	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	103	8,990	6
	16k	106	9,169	6
	24k	106	9,222	0
	128k	104	9,131	0
	512k	108	9,375	0
	1448k	107	9,353	0
P=5	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	98.9	9,041	0
	16k	102	9,276	13
	24k	105	9,334	20
	128k	102	9,409	0
	512k	102	9,378	13
	1448k	108	9,711	0
P=10	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	103	9,266	20
	16k	99.8	9,236	20
	24k	98.2	9,439	26
	128k	88.1	9,537	6
	512k	89.3	9,682	13
	1448k	104	9,790	13
P=20	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	89.5	8,547	13
	16k	80	9,353	13
	24k	86.6	9,619	6
	128k	98.9	9,819	13
	512k	84.4	9,869	20
	1448k	99.4	10,169	13
P=50	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	68.4	8,449	13
	16k	53.6	8,726	6
	24k	47.2	9,198	6

	128k	42	9,114	6
	512k	77.6	9,461	20
	1448k	79.3	11,621	13
P=100	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	32.6	8,642	20
	16k	14.8	9,175	0
	24k	4.83	9,156	0
	128k	5.5	9,807	6
	512k	39.6	10,581	13
	1448k	48.3	13,016	0

表 10 iperf 網路效能測試數據-傳統縱深防禦(無 IPS)

P=1	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	909	78509	0
	16k	912	78765	0
	24k	934	80640	0
	128k	922	79625	0
	512k	913	78873	0
	1448k	914	78951	0
P=5	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	939	81237	0
	16k	937	81039	0
	24k	939	81223	0
	128k	936	80964	0
	512k	939	81306	0
	1448k	935	81089	0
P=10	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	940	81461	0
	16k	938	81275	0
	24k	938	81248	0
	128k	940	81503	0
	512k	937	81278	0
	1448k	940	81645	0
P=20	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	940	81699	0
	16k	937	81554	0

	24k	937	81697	0
	128k	937	81506	0
	512k	937	81650	0
	1448k	937	82107	0
P=50	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	828	82412	0
	16k	817	82377	0
	24k	930	82419	0
	128k	937	82499	0
	512k	819	82866	0
	1448k	816	83507	0
P=100	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	817	83499	0
	16k	934	83491	0
	24k	936	83518	0
	128k	939	83747	0
	512k	817	84520	0
	1448k	822	86321	0

表 11 iperf 網路效能測試數據-傳統縱深防禦(有 IPS)

P=1	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	119	10309	13
	16k	124	10732	6
	24k	122	10556	0
	128k	127	11015	0
	512k	129	11217	0
	1448k	129	11241	0
P=5	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	100	10408	6
	16k	118	10551	6
	24k	119	10581	13
	128k	118	10555	6
	512k	118	10547	0
	1448k	116	10853	6
P=10	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	116	10423	6

	16k	112	10598	6
	24k	115	10541	0
	128k	118	10682	0
	512k	120	10973	0
	1448k	117	10935	0
P=20	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	97.8	10445	26
	16k	95.4	10706	13
	24k	115	10703	6
	128k	120	11122	6
	512k	113	11050	13
	1448k	117	11651	20
P=50	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	101	10873	13
	16k	110	10992	0
	24k	101	11096	20
	128k	77.6	10762	6
	512k	67.7	11857	13
	1448k	110	12751	6
P=100	Payload Size	Throughput(Mbits/sec)	PPS	packet loss (%)
	8k	39.8	10756	13
	16k	11.7	7715	6
	24k	30.7	9910	6
	128k	22.2	11134	6
	512k	35.1	12878	20
	1448k	76.2	14957	0

附　錄　三

附錄三為 httperf 根據本研究要求的 4 種架構(6 種情況)所測試出來的數據，以表 12 為例，此為無防禦設備之情況，只有 PGHost 及 Web Server 兩台主機直接測試的數據，我們可以看到不論是測試 15k 或 45k 的檔案大小時，在每秒 100 個 requests、每秒 300 個 requests、每秒 500 個 requests 到每秒 700 個 reqeusts，其 Response Time 及 Request/Replies 錯誤率都沒什麼太大的變化，而每秒封包數則呈現等比成長。其餘情況如表 13 至表 17 所示。

表 12 httperf 網路效能測試數據-無防禦設備

File size	100 requests/sec			300 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	1130	655	PPS	3390	1969
	Response Time(ms)	0.3		Response Time(ms)	0.3	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	3230	1513	PPS	9690	4521
	Response Time(ms)	0.2		Response Time(ms)	0.2	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
File size	500 requests/sec			700 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	5650	3267	PPS	7910	4602
	Response Time(ms)	0.3		Response Time(ms)	0.3	

	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	16150	7577	PPS	22611	10586
	Response Time(ms)	0.2		Response Time(ms)	0.2	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	

表 13 httperf 網路效能測試數據-整合威脅管理系統

File size	100 requests/sec			300 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	1130	660	PPS	3390	1962
	Response Time(ms)	1.5		Response Time(ms)	1.5	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	3230	1499	PPS	9690	4516
	Response Time(ms)	2.0		Response Time(ms)	2.0	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
File size	500 requests/sec			700 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	5650	3254	PPS	7910	4591
	Response Time(ms)	1.5		Response Time(ms)	1.7	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	16150	7553	PPS	22610	10502
	Response Time(ms)	3.1		Response Time(ms)	505.5	

	Request/Replies Error Rate	0.00	Request/Replies Error Rate	0.02

表 14 httperf 網路效能測試數據-虛擬化縱深防禦(無 IPS)

File size	100 requests/sec			300 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	1130	690	PPS	3390	2070
	Response Time(ms)	1.0		Response Time(ms)	1.0	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	3230	1772	PPS	9690	4765
	Response Time(ms)	1.1		Response Time(ms)	0.8	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
File size	500 requests/sec			700 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	5650	3450	PPS	7910	5083
	Response Time(ms)	1.2		Response Time(ms)	3	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	16150	7916	PPS	9191	6175
	Response Time(ms)	1.0		Response Time(ms)	465.5	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.14	

表 15 httperf 網路效能測試數據-虛擬化縱深防禦(有 IPS)

File size	100 requests/sec			300 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	1130	690	PPS	3390	2070
	Response Time(ms)	1.6		Response Time(ms)	1.4	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	3230	1768	PPS	6785	4354
	Response Time(ms)	1.6		Response Time(ms)	847.9	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.05	
File size	500 requests/sec			700 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	5650	3449	PPS	7910	4831
	Response Time(ms)	1.7		Response Time(ms)	3.2	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	6592	4220	PPS	6545	4191
	Response Time(ms)	907.8		Response Time(ms)	973.7	
	Request/Replies Error Rate	0.15		Request/Replies Error Rate	0.24	

表 16 httperf 網路效能測試數據-傳統縱深防禦(無 IPS)

File size	100 requests/sec			300 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	1130	600	PPS	3390	1974

	Response Time(ms)	0.8		Response Time(ms)	0.8	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	3230	1529	PPS	9690	3948
	Response Time(ms)	0.8		Response Time(ms)	0.8	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
File size	500 requests/sec			700 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	5650	2955	PPS	7910	4127
	Response Time(ms)	0.8		Response Time(ms)	0.8	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	16150	7524	PPS	22610	9033
	Response Time(ms)	0.9		Response Time(ms)	1.1	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	

表 17 httperf 網路效能測試數據-傳統縱深防禦(有 IPS)

File size	100 requests/sec			300 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	1130	689	PPS	3390	2068
	Response Time(ms)	1.0		Response Time(ms)	1.0	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	3230	1775	PPS	9269	5741
	Response	1.0		Response	234.8	

	Time(ms)			Time(ms)		
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.01	
File size	500 requests/sec			700 requests/sec		
	Avg.	Client (input)	Client (output)	Avg.	Client (input)	Client (output)
15k	PPS	5650	3447	PPS	7912	4837
	Response Time(ms)	1.4		Response Time(ms)	2.7	
	Request/Replies Error Rate	0.00		Request/Replies Error Rate	0.00	
45k	PPS	9648	6116	PPS	9945	6403
	Response Time(ms)	627.3		Response Time(ms)	601.9	
	Request/Replies Error Rate	0.07		Request/Replies Error Rate	0.13