A Secure PC-Based Architecture for Remote Server Management

林鳳銘　　張景堯　　李蔡彥
國立政治大學 電子計算機中心
{l96in, jychang, li}@nccu.edu.tw

摘要

網際網路不斷發展，每天都有各式為數不少的新伺服器上線，提供各種不同的服務。伺服器提供的服務也許不同，但對每一部伺服器而言，共同的例行工作就是伺服器的管理。一般而言，伺服器的管理者必須透過網路或者站在主控台(console)前進行管理的工作；如何簡化程序且安全地管理伺服器，是一項十分重要的課題。在本文中，我們改變原有透過網路或在主控台前直接管理伺服器的方法，而提出另一種透過 PC-based 防火牆與序列埠主控台管理系統(Serial Console Management)來管理伺服器的架構。這個架構類似內含防火牆的 IP-based KVM，但所花費的成本僅需要一般等級的 PC 伺服器。透過這種架構，管理者不必擔心新安裝好的伺服器一旦接上網路後，就立刻中毒或者被入侵；管理者也不一定要站在主控台前來管理伺服器，而可以在伺服器上線後安全地對新系統執行修補(patch) 或者 Windows update 的動作，以確保伺服器的安全狀態。為了瞭解進出伺服器的資訊，我們也同時在所提出架構中的防火牆上，建立了一個簡單而且可以輸出 Netflow 資訊的方式，讓我們可以透過網路流通的資訊來進一步掌握伺服器的狀態。

關鍵詞：主控台、序列埠、修補程式(patch)、IP-Based KVM。

Abstract

With the continuous development of the Internet, new network services are brought on-line every day. Despite the service contents provided by the servers are different, a common routine task for every server is daily system administration. Generally speaking, the administrator of a server uses the network or stands in front of the machine to remotely or directly perform the tasks of system administration. It is an important issue on how to provide a convenient management scheme without sacrificing system security. In this paper, we have proposed a new economical architecture with a PC-based firewall and a serial console management scheme to provide the service of remote server management. The architecture is similar to IP-based KVM but the cost is much less. By the use of this architecture, the system administrator does not need to worry about that a server may be infected or intruded before necessary security patches are applied. The system administrators can remotely bring a system on-line and then apply system patches or windows updates without putting the system security into jeopardy. In order to understand the information flowing in and out of the server, we have also implemented a netflow-based monitoring system to proactively detect network anomalies.

Keywords: Server Console, Serial Port, Patch, IP-Based KVM.

1. 前言

由於網際網路已經逐漸成為日常生活的一部份，已有越來越多的資訊服務透過網際網路提供。目前這些網路服務，仍多以主從式（Client-Server）的架構存在。為提供穩定的服務，伺服器的安全性就顯得格外的重要。影響伺服器安全性的因素很多，其中最基本的就是主機存取控制（Access Control）。一般而言，管理者通常會透過 ssh/telnet/http 等遠端方式來管理伺服器；而這種管理方式需要伺服器除了執行提供此管理服務的 daemons （如 sshd/inetd(xinetd)/httpd）及相對應的 ports 外，有時還需執行額外的 daemon 及開啟相對應的 port ，以利管理者透過網路管理伺服器。伺服器上所需要執行的管理 daemon 數目越多，需要開啟相對應的 port 也就越多；對伺服器而言，所需承擔的網路風險自然也就越大。

為了簡化伺服器的管理，有許多的網路廠商已經開發出各種不同的 IP-Based KVM 系統，來提供管理者透過遠端(網際網路)來管理伺服器[1][2]。這類型廠商所提供的管理方式，需要管理者購買一些相關的網路設備來配合管理。有時為了提供多部主機的不同管理者可以同時管理伺服器，還需要額外購買帳號管控伺服器以利帳號管理；而且這些設備的價格昂貴，往往需要數十萬元。

另外，雖然使用這些設備的優點是可以透過網路來管理伺服器，但他們多沒有提供防火牆的功能。如果伺服器是剛安裝好的系統，尚未執行相關的安全性補強，則在連上網路的同時，則立即可能受到安全的威脅。由於目前網際網路上的病毒，如蠕蟲 (worm)，都已經自動化，且善於偽裝、隱蔽欺敵。一旦伺服器上執行的網路相關 daemon 有漏洞或作業系統含有 Vulnerability，病毒、蠕蟲或駭客就會自動找上這部伺服器。如此一來雖可以遠端

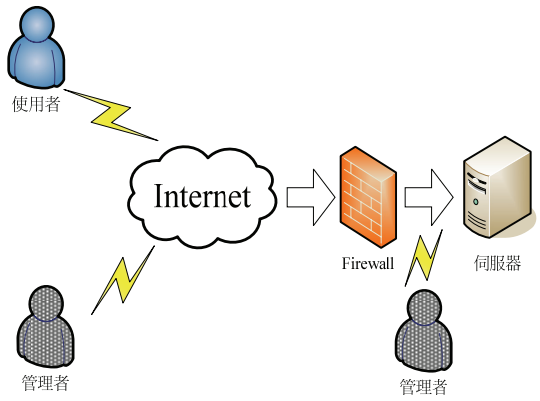

圖 1 常見伺服器的管理方式

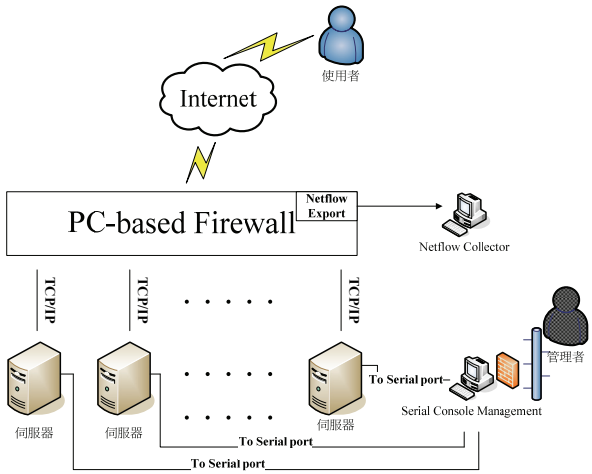

圖 2 以 PC-Based 防火牆與序列埠來管理伺服器的架構

管理，不過一旦連接上網路後，這部伺服器很可能就馬上中毒或遭入侵了。解決這個問題的方法之一，是將此伺服器安置在防火牆之後上網，以阻隔病毒或者入侵者的攻擊。雖然也有一些提供防火牆設備的廠商[3]，但這又需要購買一些所費不貲的資安設備，同時改變伺服器的連接方式，才能達到系統安全的效果。

圖 1 說明了管理者常用的的伺服器管理方法，大約可以分為有兩種。第一種是管理者可以經由網際網路來管理伺服器。對這種方式而言，管理的彈性最大，也最容易管理；因為不論管理者位於什麼地方，只要可以連上網路，都能透過網路來管理伺服器。不過，由於伺服器必須在網路上提供管理的 port，這種管理方式的風險也比較大。網路上的任何使用者都可以透過網路連結到管理的 port 上，即使伺服器管理者為這個管理 port 建立存取控制，但由於存取管控是在執行管理服務 daemon 時進行控制，因此一旦在執行於這個 port 上用來管理的 daemon 有安全性漏洞時，網路上的病毒或 cracker 就可以輕易地利用這個漏洞入侵這部伺服器。

第二種管理方式是管理者只能在防火牆內來進行伺服器的管理。以這種方式來管理伺服器，防火牆後的這一段管理用網路的安全性就顯得十分重要。如果網路管理者以實體的方式獨立區分這一段管理網路，雖可以提升伺服器的安全性，不過卻犧牲了便利性。當管理者需要管理伺服器時，就必須實體找到位於防火牆內的機器，方能進行管理。另一方面，如果網路管理者並沒有以實體的方式獨立區分管理網路，則也同樣會發生與第一種管理方式一樣的問題。對於防火牆後的管理方式，只要開啟了額外管理用的 port 與 daemon，都存在著風險。

為了克服傳統管理伺服器所遭遇到的缺點，同時兼顧安全與遠端管理，我們提出了一個以 PC-Based 防火牆與序列埠來管理伺服器的架構。防火牆可以用來避免新安裝或有安全性漏洞的主機一連接上網路就立刻被病毒或 cracker 入侵，而使用序列埠則可以用來提供另一種遠端管理伺服器的方法。

2. 系統架構

圖 2 是我們提出的系統架構摘要圖。根據圖 2，我們改變了圖 1 所說明的兩種的管控方式，將需要管理的伺服器，一律放到防火牆後面，同時對需要提供服務的伺服器只開啟該服務所需要的 port，其餘的 port 一律關閉。另外，我們在防火牆上安裝可以輸出 Netflow[4]資訊的 daemon 來即時將網路的流通資訊傳送出去，同時架設一部 Netflow Collector 來收集防火牆所輸出的 Netflow 資訊。至於伺服器的管理則交由圖 2 中位於防火牆後的 Serial Console Management 主機來進行。為了管理 Serial Console Management 主機，我們在路由器上虛擬區分這一段網路，並建立適當的存取控制，只有讓少數的機器可以透過網路連接上 Serial Console Management 的主機來進行伺服器的管理。

2.1 PC-Based 防火牆架構

我們所建立的 PC-Based 防火牆，主要用來過濾傳送到到伺服器的封包。過濾的規則預設為阻擋所有傳送到新加入伺服器的封包，同時允許新加入伺服器傳送封包到防火牆外，並建立連線狀態監測。當新加入伺服器欲傳送封包到防火牆外的網站時，防火牆會建立所欲連結網站的連結狀態，同時暫時允許被連結網站的資訊，可以傳送到防火牆內新加入的伺服器上。新加入的伺服器就可以在這個時候進行 patch 或 Windows update。如此一來，新安裝好的伺服器，連接到防火牆後面時，並不會發生一連結上網就發生被入侵或中毒的情形。

另外，根據伺服器的服務類型，我們在防火牆上開啟封包的轉送工作。對已經執行 patch 或 Windows update，同時做好了封包轉送設定的伺服

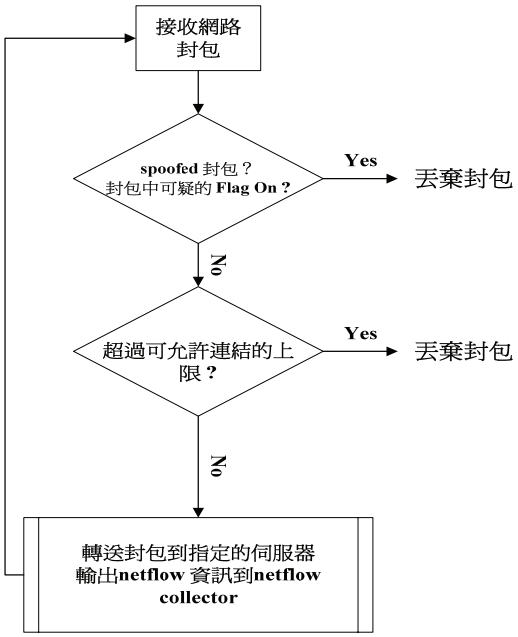

圖 3 PC-Based 防火牆封包過濾的規則

器來說，防火牆過濾封包的機制如圖 3 所示。當防火牆接收到網路傳送的封包時，會先檢查封包的 header 看看是否有異常的旗標(flag) 被設定，如果有檢測到異常旗標設定的封包就予以丟棄。如果是 header 正常的封包就繼續檢測是否達到我們所設定的允許連結的速率（舉例來說，傳送到某一部伺服器的資訊流在每 5 秒內不能超過 250 個）；如果超過，這一類的封包也將被丟棄。如果以上的過濾條件都吻合時，就會根據封包所要送達的目的地伺服器做轉送的工作。由於所有傳送給伺服器的封包都會先經過防火牆，因此我們在防火牆上建立了 Netflow 輸出系統，來將所有的封包輸出給 Netflow Collector 伺服器，然後再分析所收到的 Netflow[5] 資訊，檢查是否有異常的流量進出，以確保伺服器安全。

對於接收 Netflow 資訊的 Netflow Collector ，我們分析了伺服器傳送與接收的資訊流(flow)、封包數量(packets)與資料量(octets)。圖 4 所呈現為流通到某一部伺服器資訊流的數量變化情形。我們可以根據資訊流的變化，來簡單觀測伺服器的負載情形。我們也可以根據伺服器傳送與接收的資料量大小，來判別伺服器是否遭到誤用，成為資料轉送中心，不斷傳送大量的資料到網路中。由於所有有關伺服器的進出資訊都必須經過我們所建立的 PC-Based 防火牆，因此可以透過此防火牆機制，將所有進出資訊全部透 Netflow exporter 來傳送到圖 2 中的 Netflow Collector。我們除了可以分析上述提到的資訊流量外，也可以透過此介面即時處理一些有關伺服器的非正常連結行為及未經授權的連結狀況。

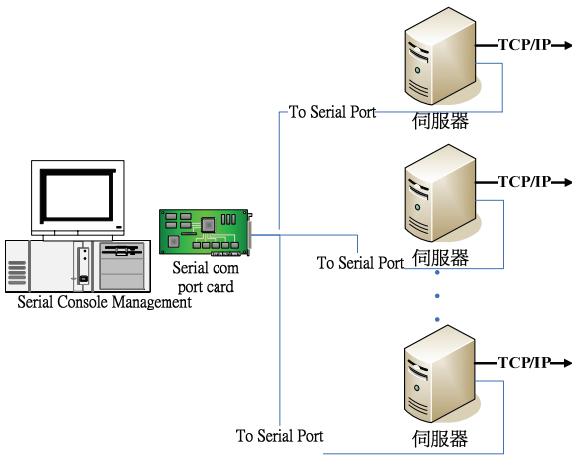

圖 4 序列埠主控台管理系統架構

2.2 序列埠主控台管理系統架構

我們所提出的序列埠主控台管理系統(Serial Console Management)，所包含的元件是一部一般 PC 與一張（視需要可以動態增加）4 port 或 8 port 的序列埠卡。需要被管理的伺服器必須要有序列輸出埠。對一般 x86 伺服器而言，大多具有序列輸出埠；對工作站級的伺服器，如 Sun/SGI 來說，model 比較舊的機器有 serial port 可以連接，而比較新的 sun 工作站則有 Serial MGT 埠可以使用。如圖 4 所示，被管理的伺服器以序列埠與序列埠主控台管理系統相連接。當需要管理伺服器時可以直接由序列埠進行管理，管理的方式與管理者親自站在主控台前管理一樣，即使在伺服器尚未連接網路的情況下，也可以進行遠端管理。

3. 系統實做

對於上一節所提出的架構，我們將在本節中以三部等級不同的 PC 來分別實作 PC-Based 防火牆、序列埠主控台管理系統與 Netflow Collector。以下是這些系統的實作方式。

對於 PC-Based 的防火牆，由於許多的伺服器將會連接到他的後面，同時也執行即時的 Netflow 輸出動作，其系統的負載將比較重。因此，我們以性能較佳的 Intel Xeon 伺服器來架設，而其上的防火牆系統，則是用 Packet Filter [6]來實作。序列埠主控台管理系統，由於不需要大量的運算與強大的網路功能，因此我們可以用一部 Intel P-III 等級的 PC 來實作即可。序列埠主控台的管理是透過在邏輯上切開的獨立網段，以加密的 ssh 來管控。對於所需要的序列埠控制卡，我們則是使用以 SUN1889 晶片為基礎的 SUNIX 4066R 8 port 序列卡。當有伺服器連接時，需要將伺服器原本預設輸出到監視器

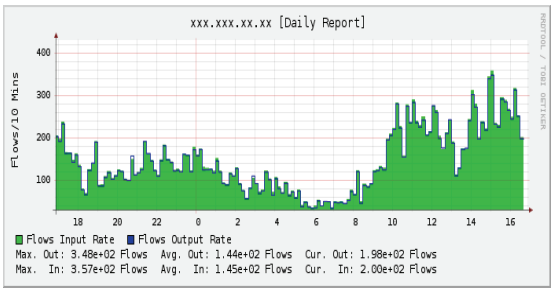

圖 5 伺服器傳送接收的資訊流

的設定作更改。這對 UNIX 相關的系統，如 Linux/*BSD 而言可以透過修改設定檔或重新編譯系統核心來達成，設定好後之後，就可以用序列埠來管理這部伺服器。至於 Netflow Collector 則根據伺服器的流量大小我們選用了一部較舊但適當的 Intel Xeon 伺服器來當 Netflow Collector。我們使用 flow-tools[7]實做 Netflow 接收器。對於所接收到的 Netflow 資訊，我們撰寫了一些分析 Netflow 的程式，同時以圖表的方式來呈現分析後的結果，圖 5 便是其中一個例子。

目前大部分的機房都有發電機與不斷電系統，可以在市電停電時提供伺服器電源。雖然機房能夠提供穩定的電源，不過為了能夠進行遠端管理電源，我們額外購置了 WTI 遠端電源管理設備[8]。藉由這個設備，我們可以遠端控制的方式，對伺服器的電源進行開與關的動作。為了避免萬一發生網路設備當掉而無法透過網路連接序列埠主控台管理系統，我們用一部已經不再使用的老舊 cisco2522，將他的 aux port 與序列埠主控台管理系統的 serial port 相連接，同時將 cisco 2522 的 Ethernet port 連接到主幹路由器，我們就可以透過網路與 cisco 2522 的 aux port 來控制序列埠主控台管理系統，詳細的系統架構如圖 6。

我們所提出的系統已經上線提供正常服務。以 2005/08/14 為例，此系統在國立政治大學電算中心的負載約在為 15.16 flows/sec，同時我們自 PC-Based 防火牆的 log 中找出一小時內被阻擋的 flow 數量為：29471 次，如此計算可以知道 flow 的阻擋率約為 54%。

4. 結論

對系統管理者而言，系統一旦上線提供服務後，為使系統能提供正常而穩定的服務，定期例行的系統管理與維護工作就顯得相當重要。對管理者而言，許多的管理工作所執行的命令都需要有管理者的權限方能執行；如何安全與便利地執行遠端管理便是一項挑戰。在本文中，我們以廉價的 PC 伺服器，建立一個以 PC-Based 防火牆及序列埠主控台管理系統(Serial Console Management)為主的架

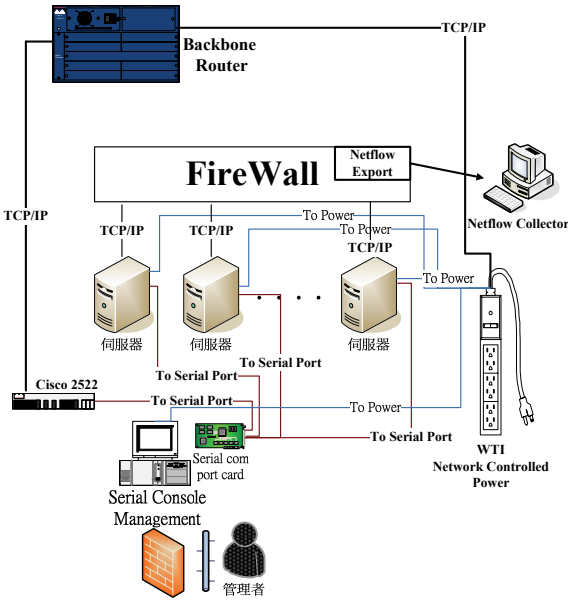

圖 6 PC-Based 防火牆與序列埠伺服器管理系統的詳細架構圖

構來管理網路服務伺服器。管理者可以透過遠端執行命令來安全而方便地管理伺服器。在這個架構下，管理者不必擔心新購入或新安裝好的伺服器，在連接網路時很容易中毒或者被入侵，也不一定要站在主控台前來管理伺服器。連接到 PC-Based 防火牆後的伺服器還有一項優點，是當伺服器發生需要停機檢修的狀況時，如果備援主機已經連接在防火牆之後，則可以動態地重組防火牆上的設定[9]，使得備援主機可以及時備援，對於所提供的服務將不會發生服務中斷的情形。由於所有傳送到伺服器的資訊都必須經過 PC-Based 防火牆，因此為了避免發生 single point of failure 的問題，我們可以架設另一備援防火牆來解決[10]。由於本文所提出的遠端管理方法是透過指令來方便管理者以不必親自到機房的方式就能進行遠端管理。對於一些僅能透過圖形介面管理的系統（如 WINDOWS），並不能以指令進行遠端管理；不過仍能連接到防火牆後定期執行 WINDOWS 相關的 updates，以避免病毒與 cracker 的入侵。

參考文獻

[1] Avocent, http://www.avocent.com

[2] Raritan, http://www.raritan.com.tw

[3] Checkpoint, http://www.checkpoint.com/

[4] Cisco, http://www.cisco.com

[5] P. Barford and D. Plonka. “Characteristics of Network Traffic Flow Anomalies”, in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, NOV. 2001.

[6] R. McBride, “Introduction to PF”, BSDCAN 04 in

Ottawa, Canada, May 2004.

[7] Flow-tools information, http://www.splintered.net/sw/flow-tools/

[8] Western Telematic Inc.,http://www.wti.com/

[9] 林鳳銘、吳守豪、李蔡彥, “PC-Based 伺服器 Fail-Over 系統之建置”, in *Proceedings of TA-NET 2004 Conference*, pp. 208-211, 2004.

[10] http://www.openbsd.org/faq/pf/carp.html