國 立 交 通 大 學

資 訊 工 程 學 系

碩 士 論 文

預防擬態攻擊之入侵偵測防禦系統

An Intrusion Prevention System against Mimicry Attacks

指導教授：蔡文能　教授

研 究 生：鄭光宏

中 華 民 國 九 十 四 年 七 月

預防擬態攻擊之入侵偵測防禦系統

An Intrusion Prevention System against Mimicry Attacks

指導教授：蔡文能　　Advisors：Wen-Nung Tsai

研 究 生：鄭光宏　　Student：Kuang-Hung Cheng

國 立 交 通 大 學

資 訊 工 程 研 究 所

碩 士 論 文

A Thesis Submitted to

Institute of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Computer Science and Information Engineering

July 2005

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 四 年 七 月

預防擬態攻擊之入侵偵測防禦系統

學生：鄭光宏　　指導教授：蔡文能 教授

國立交通大學資訊工程學系（研究所）碩士班

摘　　要

隨著硬體與網際網路技術的發達，各種資訊領域之應用紛紛在網路上出現，然而在這開放的網路環境中，卻充斥著許多惡意的攻擊者。許多學者專家設計出不同的入侵偵測技術，然而攻擊者則是針對這些技術的弱點加以攻擊、或是規避其偵測，更增加了設計入侵偵測防禦技術的難度。我們分析比較各種入侵偵測防禦技術和其弱點，並提出解決一部份問題的方法。

本篇論文以攔截系統呼叫(System Call)技術爲基礎，設計出一套即時的入侵偵測與防禦系統，AMA-IPS(An Intrusion Prevention System against Mimicry Attacks)。使用者可以圖形介面，狀態轉換之方式描述攻擊樣板。我們針對傳統入侵偵測技術上的弱點進行改進，提高了入侵偵測防禦的準確度。另外，我們以人體免疫系統模型(Immunity Model)來檢驗攻擊樣板的精確性，以減少誤判。本系統可攔截所有應用程式所請求的系統呼叫，基於使用者所定義之攻擊樣板，持續追蹤可疑的應用程式，於攻擊動作尚未成功前就終止其執行，以提供精確，即時，有效的防護網。

An Intrusion Prevention System against Mimicry Attacks

Student：Kuang-Hung Cheng **Advisors：Dr. Wen-Nung Tsai**

Institute of Computer Science and Information Engineering
National Chiao Tung University

ABSTRACT

With the development of the hardware and Internet technologies, there are lots of applications available on the Internet. However, there are always hostile assailants in the open network environment. Though many different intrusion detection techniques had been developed, assailants can always attack against the weakness on these techniques, and try to evade from IDS detection.

Based on system call interception technique, we develop a real-time intrusion detection and prevention system, called AMA-IPS (An Intrusion Prevention System against Mimicry Attacks). In this system, users can describe the model of attacking, through a GUI interface, in the form of state changes. We integrated the immunity-based techniques into the state-based IPS to detect mimicry attacks and thus improve the detection accuracy of the IPS. In addition, we examine penetration pattern's accuracy with the human immune system model, and thus reduce false positive. This system intercepts every system call invoked by an application program and tries to match any penetration pattern. Once there is an evidence showing some penetration is undertaking, the system can terminate the penetration process before injury.

致　　謝

經過兩年的努力，終於順利完成了我的畢業論文，在此感謝所有人的幫忙。首先要感謝的是我的指導教授—蔡文能教授，在碩士班的兩年，他給我許多方面的指導，讓我受益良多，在此十分的感謝。接著要感謝的是實驗室的蔡宗易學長，感謝他不辭辛勞的指引我研究的方向，非常感謝。接著感謝實驗室的同學們，大家一起工作、學習，互相幫忙，才能度過難關，讓我有個充實的碩士班生活。

許多的感覺無法用言語形容，對於上面提到的所有人，在此致上我最深的謝意，謝謝大家。

目　　錄

表　目　錄

圖　　目　　錄

第一章、緒論

回顧網際網路發展的歷史，從最早期ARPANET 的建立，到之後Internet 的普及、各種區域網路標準的競爭，可以觀察到其中最重要的議題，就是解決連結性(connectivity)的問題。如今，這個議題已大致獲得解決，現今幾乎所有政府機關、學校、企業都有自己的網站，網際網路已經與每個人的生活息息相關，密不可分。

當連結性問題逐步獲得解決之際，安全性問題卻逐漸浮現。近年來安全事件不斷發生，不僅造成企業巨大損失，更直接危及消費者使用網際網路服務的信心。當今眾多的網路應用程式，如電子郵件客戶端(email client)、網頁伺服器(web server)、檔案傳輸伺服器(ftp server)等，經常存在著一些弱點與漏洞，讓攻擊者有機會加以利用來進行破壞、甚至竊取重要資訊。再者由於軟體發展已趨成熟，軟體開發者發展程式時，往往需結合許多非自身設計的現有模組，來完成功能更複雜的軟體，這也使得程式中的錯誤或漏洞越發難以察覺。緩衝區溢位(buffer overflow)、後門(back door)、邏輯錯誤(logic error)、甚至單純的使用者設定錯誤(misconfiguration)都有可能招致電腦駭客的攻擊。

爲了抵抗惡意的攻擊者，資訊安全領域的學者專家們發展出了許多防禦工具，如防火牆(firewalls)、防毒系統(anti-virus)、入侵偵測系統(Intrusion Detection System ,IDS)，希望能夠偵測並預防攻擊行爲。然而，無論這些防禦工具設計再複雜、功能再強大，駭客對其徹底研究以後，總是能找出特殊的方法來破解它，並繞過這些防禦工具進行攻擊。當然，防禦工具被破解後，資安專家也會立即找尋應對方法，將防禦工具進行強化，以抵擋新型的攻擊。因此，長久以來，資訊安全領域中許多研究,，便是在資安專家與駭客之間的互相競爭中產生出來的。

入侵偵測系統是其中一項重要的安全性防禦工具。入侵偵測技術依照偵測的模式可分成「不當行爲偵測(misuse detection)」和「異常偵測(anomaly detection)」。不當行爲偵測是最常用於IDS上的偵測方式，又稱爲「特徵比對(signature-based detection)」，是利用已知的攻擊事件建立出各種攻擊特徵資料庫，當偵測到的行

爲與資料庫中某個特徵相符，就會將它視爲入侵。此種做法的優點是誤判(將正常視爲異常)率較低，但缺點是偵測率也較低，因爲新型態的入侵若未建立在特徵資料庫中，便無法偵測出來。而異常偵測則是建立正面行爲模式，即在系統建立正常行爲模式資料庫，將所偵測到的行爲與正常行爲模式進行比較，若差異甚大便視爲異常行爲。此種做法的優點是偵測率高，可以偵測到新型態的入侵，但是同時誤判率也高，因爲使用者的行爲經常在變，精確定義何謂「正常行爲」實非一件容易的工作。

本篇論文採用不當行爲偵測的方法，基於STAT(State Transition Analysis Tool)[14][20]，我們設計出一個即時入侵偵測防禦系統，AMA-IPS(An Intrusion Prevention System against Mimicry Attacks)，將其實作於Linux系統上。我們將討論現有的IDS系統上，不論是不當行爲偵測或是異常偵測，會有的弱點與規避其偵測的方式，並在我們的系統中提出解決的方法。

1.1 動機與目的

入侵偵測技術行之已久，很多文獻對IDS都不斷提出改進的方法，對提高偵測率和降低誤判率都有不少貢獻。然而駭客的攻擊方式不斷翻新，加上IDS作法公開地被廣泛討論[31]，許多研究也提出破解IDS防禦的方法，如擬態攻擊(Mimicry Attacks)[4]。大部分攻擊行爲中，攻擊者的目的莫過於取得使用作業系統資源的特權，進而進行破壞或竊取資訊。而向作業系統取得資源的唯一方法，便是透過系統呼叫(system calls)。因此，我們以攔截系統呼叫爲出發點，透過監督程式執行時的系統呼叫來防禦攻擊，並且以STAT爲基礎，利用以狀態轉移來定義攻擊的優點，使我們設計出來AMA-IPS可以克服現有IDS系統的缺點，使之能夠抵抗[4]中對IDS的攻擊。

1.2 論文架構

本篇論文的組織架構如下：『第二章、背景知識』將介紹學術上被提出的各種入侵偵測研究，實作上攔截系統呼叫的技術，以及本論文參考到的人體免疫系統之概念。『第三章、相關研究』介紹幾種現有的入侵偵測系統，以及在其上可行的一些攻擊方式。『第四章、系統需求與設計方法』說明本系統要達成的三大

目標與設計的方法。『第五章、系統架構』將詳細介紹本系統的組成元件與模組。『第六章、實驗與評估』顯示運用本系統防禦攻擊之實驗結果與優劣比較。『第七章、討論與結論』介紹論文結論與論文未來的擴展可能性以及需要增進的部份。

第二章、背景知識

在這個章節，我們將介紹本篇論文的相關背景知識。首先，對入侵偵測(Intrusion Detection System, IDS)與入侵預防(Intrusion Prevention System, IPS)技術作一個概括性的簡介。2.2中，介紹幾種入侵偵測技術的設計方法。爲了在核心層(kernel)實作入侵預防，我們必須在核心層攔截系統呼叫，把系統設計成可載入核心模組(Loadable Kernel Module)，這些將會在2.3中一一介紹。2.4中，介紹人體免疫系統之概念，以及從中得到設計入侵預防系統的啓發。

2.1 入侵偵測與入侵預防技術簡介

由於資訊科技(Information Technology ,IT)和網際網路的快速發展，參與網際網路的人數日益增加，根據資策會的調查統計，至2004年6月底止，台灣區的上網人口已達892萬人(如圖1 所示)，網際網路的組成份子愈來愈複雜，如何在這個混亂的大環境中抵禦入侵者，確保自身的安全，無論對企業或是對個人來說，都是相當重要的議題。

網際網路中的入侵方式，通常可以分成下列三步驟：收集資訊、入侵和入侵後的處理(如圖2 所示)。收集資訊是指盡量收集有關目標主機的所有資訊，例如主機網址、主機開放的服務、主機內的使用者帳號(User ID)、使用者密碼(User Password)、甚至是系統管理者密碼。接著是入侵動作，可以是直接以前一步驟取得的系統管理者密碼進入主機，或間接利用漏洞進入。成功入侵主機後，接著是最後的處理動作，包括清除系統中可能形成有關入侵行爲的記錄，以避免留下犯罪證據，也可能執行後門程式以方便下次的進入。

入侵行爲從步驟一收集目標主機的資訊開始，目標主機便有機會透過分析網路封包，或是分析系統中若干關鍵點的訊息，來檢查網路或系統中是否存在違反安全策略的行爲和被攻擊的跡象。而進行入侵偵測的軟體或與軟硬體的組合便是

IDS。

圖 1 台灣經常上網人口成長情況

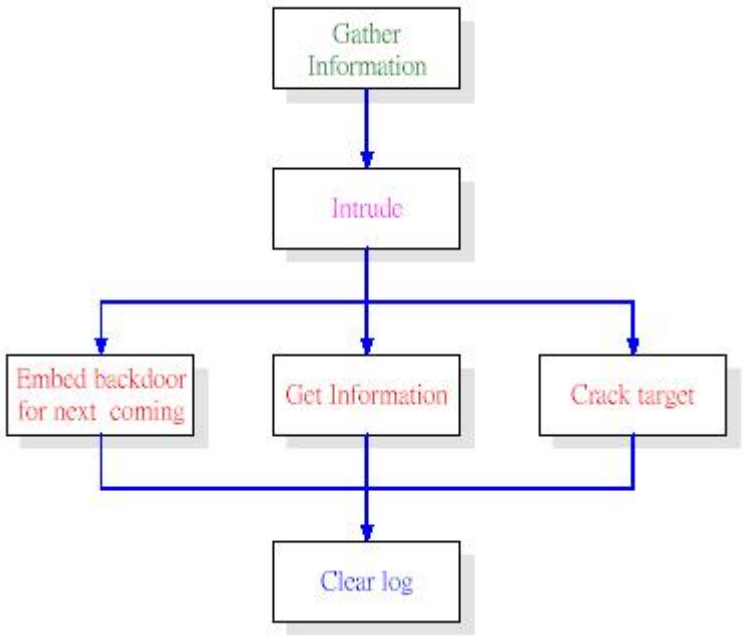


圖 2 入侵步驟示意圖

IDS的主要功能包括檢測並分析使用者在網路中的活動，辨識已知的攻擊行爲，統計分析異常行爲，查核系統配置和程式漏洞，評估系統關鍵資源檔案的完

整性，管理作業系統記錄檔，識別違反安全策略的使用者活動等。

IDS依照其偵測的範圍可以分成「網路型入侵偵測(Network-based IDS)，簡稱NIDS」和「主機型入侵偵測(Host-based IDS)，簡稱HIDS」，各有其設置考量與優缺點。NIDS大多在網路入口佈署，監看一個區域網路。NIDS的優點是成本較低，僅需在一台主機上設置IDS即可監看整個網域，免除每台主機都要安裝IDS的成本。NIDS可以偵測出阻斷服務(Dos)或通訊埠掃描(Port Scan)等攻擊，但NIDS通常僅對網路封包的標頭(header)進行過濾分析，與標頭無關的攻擊方式則無法偵測出來。HIDS常佈署於重要的伺服器或主機上，針對系統上的重要檔案與資源進行監控，或是對使用者的行爲進行監督，一旦符合入侵規則便發出警告。HIDS可以偵測到一些NIDS偵測不到的入侵，例如駭客在受害主機的一些行爲、種植後門程式等等。然而如果一個區域網路內有多台重要主機，則需爲每台重要主機安裝HIDS，成本會提高很多。

除了依照偵測範圍來分類，在技術上，IDS依照偵測的模式可分成不當行爲偵測和異常偵測，在第一章已有概觀的介紹，而我們將在下一節對此兩種模式做更詳細的討論。

早期多數IDS系統都是被動的，不是主動性的，它們被認爲是一種檢測機制，非預防手段。IDS在偵測疑似入侵行爲時會將紀錄通報給安全專家來做決定，這需要一些時間來做考慮，因此可能在系統已經被入侵後才察覺，爲時已晚。IPS與IDS最大的不同在於IPS能夠在偵測到入侵的同時進行防禦，它們的設計旨在預先對入侵活動和攻擊性網路流量進行攔截，避免其造成任何損失，例如立刻停止該程式的執行，或是切斷網路連線等等。然而，IDS和IPS技術著重的都是在於如何辨識出攻擊，兩者其實用的技術和方法都是很類似的，而且關於IDS或是IPS的學術研究，也都是著重在如何辨識出攻擊行爲。許多被提出的IDS模型，特別是「即時」的IDS系統，實際上已經具有IPS的功能，能夠達到防禦攻擊的目的。因此，本篇論文中，不會特別強調IPS與IDS的分野，而會將其視爲同一類技術。

2.2 傳統入侵偵測技術

傳統IDS依照偵測的模式可分成不當行爲偵測和異常偵測，如圖3 所示，

其效能主要由兩個因素來決定：

(1) 資料特徵(data feature)，對於不同種類的IDS來說，要處理分析的資料也會有所不同，這些資料的特徵即代表每一使用者在連線或使用期間的行爲描述，例如基於系統呼叫的主機型入侵偵測，要處理的資料便是系統呼叫以及其參數，對於網路型入侵偵測來說，要處理的資料便是收到的封包。資料的適當選定有助於IDS對使用者的行爲分析。

(2) 分析方法與工具，其作用如下：

a.用來建立攻擊規則(Attack Rules)或行爲模式(Behavior Model)。決定資料特徵後就是依此特徵進行過濾分析，這時必須利用各種分析工具，建立對應的攻擊規則與行爲模式。

b.用來進行比對工作。當收集到現行行爲資料時，需要如同建model的過程，根據特徵將行爲轉換，轉爲與先前建立行爲模式相符的比對格式。

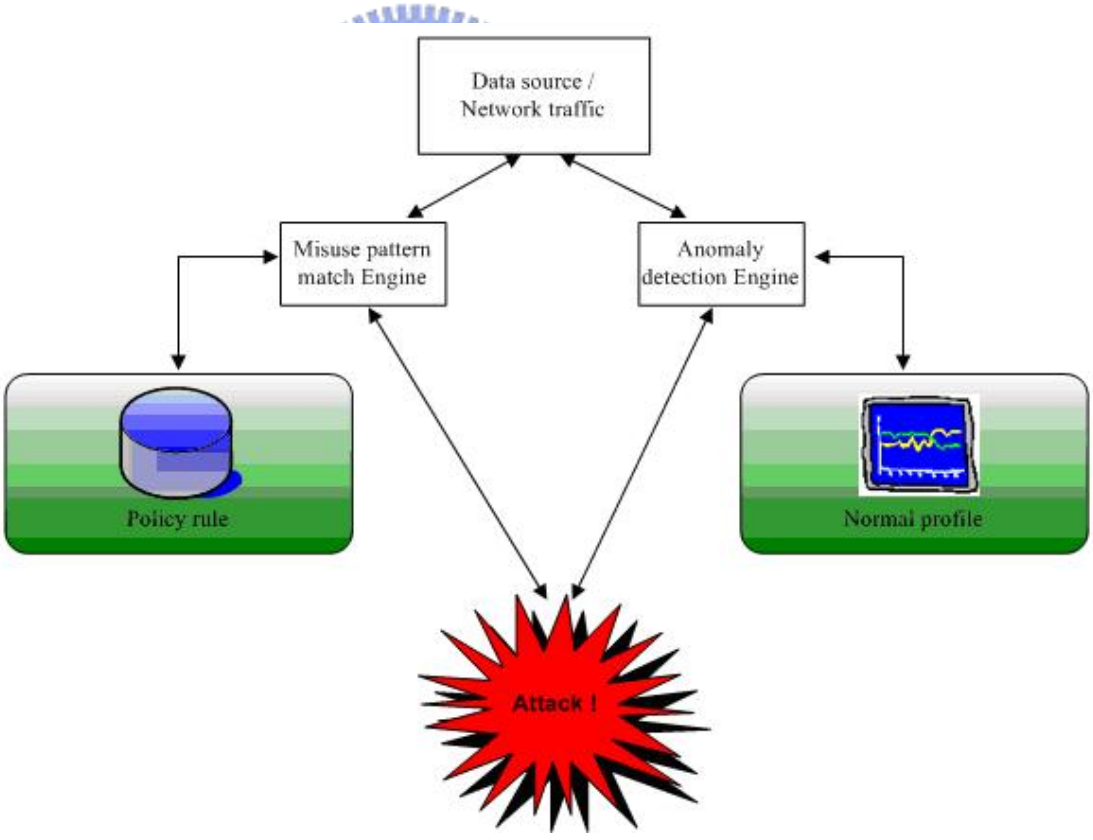


圖 3 一般 IDS 模型

在本節中將介紹這兩種模式所使用的分析方法。

2.2.1 不當行為偵測(Misuse Detection)

Misuse偵測是嘗試將已知的攻擊行爲，用各種語法描述成攻擊樣式(attack pattern)，並用這些攻擊樣式來辨認攻擊。用於辨認攻擊的分析方法常見的有三種：

(1) Rule-based分析方法-利用各種語法描述目的事件，以建立各種事件的規則庫。將行爲或是model利用說明敘述的方式表達出來，將其表示成一種規則，就像是法律一樣，只要符合rule就認定它是一種非法的行爲，一般大家所熟知的snort[19]就是屬於此分析方法的tool。此分析方法常用於misuse專家系統，如：ASAX[15]。

(2) 貝式網路(Bayesian network)分析方法-利用貝氏條件機率所延伸出來的一種學習演算法。這是運用條件機率的分析方式，貝氏定理的一個延伸應用，其原理是建立各個特徵或屬性的發生機率，再根據各屬性之間的關係計算出它們之間的條件機率關係，因而建立出完整的貝氏網路架構圖。由此可知這種分析方法最重要的關鍵在於如何建立特徵或是屬性的機率狀態分佈，如果使用的特徵或屬性無法由機率方式表現則此法便失效了。同時也可以利用許多已知的條件去推測各種可能發生的事件及其條件機率，因此貝氏網路亦有預測未知事件發生的能力。此分析方法亦常用於misuse專家系統，如：ICE[23]。

(3) Finite-state-machine分析方法-使用狀態轉移的觀點來描述目的事件的分析方法。有限狀態機的運作是用一群各自不同的狀態以及描述一些觸發事件所組成的狀態轉換，各個狀態間皆有某種關係，且這些關係就是觸發狀態轉移的事件，因此可知有限狀態機必須包含：起始狀態、輸出狀態、狀態轉變函數以及輸出函數，因此在使用此方法時要找出把資料轉化爲狀態和觸發狀態轉移的事件，才能有效的運用此分析方法。此分析方法亦常用來建立misuse專家系統，如：STAT[14][20]，USTAT[14]，STBIPW[24]。

這三種分析方法其偵測效能的好壞與所建出來的rule-base或model-base大小有關，如果使用在anomaly detection上，必須要建立出正常行爲的model，建出來的model會相當大，影響偵測的效能，因此anomaly detection較少使用這些分析方法。但若使用於misuse detection時，只需建立出已知攻擊行爲的model，所建出的attack rule並不會太多，因此對IDS的效能影響不大。

2.2.2 異常偵測(Anomaly Detection)

Anomaly偵測是去定義哪些行爲在系統裡算是正常行爲，跟這些正常行爲偏差很多的就算是一種異常行爲，因此可以偵測出未知的攻擊。分析方法常見的有兩種：

(1) 統計分析方法-利用統計方法建立規則或行爲模式(rule，behavior model)，將過去收集的資料建立個別機器的normal profile，然後進行現在的action和profile的比對，若有很大的差異，就判定此action爲異常，管理者可以對此進行加強監控。因此最重要的問題是如何建立網路上所有機器的normal profile。採用此種分析方法的代表性IDS，如：ARGUS[11]，SPADE[12]，NIDAS[3]。

(2) 類神經網路(Neural network)分析方法-是具有學習能力的演算法，經由適當訓練可使其具有辨識入侵的能力。一般類神經網路方式和用learning 的anomaly detection一樣，需要先進行正常行爲操作，蒐集後成爲類神經網路的輸入，經由training所得出來的model即爲normal model，因此它和learning之anomaly detection有著相同的缺點：一爲需要有很長的training時間，相當的耗時；二爲當我們新增一個training set的時候，整個training都要重新來過，相當不符合時效；三爲需要有很好的資料來做training，如果training資料取得不好的話，所做的normal model的誤判率會變得很大。另外類神經網路的偵測方式亦可運用在misuse detection上，也就是若training set若爲Intrusion behavior，也可達到偵測的效果。採用此種分析方法的代表性IDS，如：ACME[10]。

由於正常行爲也有很大的變動性，因此分析方法能否訓練出較好的normal behavior model，是影響anomaly偵測精確性的關鍵。

2.3 攔截系統呼叫(System call interception)

在現代的作業系統中，所有的系統資源都是由作業系統集中管理，每個想要存取系統資源的程式，都必須透過系統呼叫向作業系統提出要求，然後等待作業系統將資源分配給它。在本節中，我們將介紹攔截系統呼叫的相關技術。

2.3.1 可載入核心模組(Loadable Kernel Module)

在討論如何在核心中攔截系統呼叫之前，我們先介紹要如何在核心中加入自己需要的功能。我們可以將模組(module)視爲一個程式，但是它可以被動態載到核心裡成爲核心的一部分。載到核心裡的模組具有跟核心一樣的權力。可以存取任何核心的資料結構。雖然Linux已是一個完整的系統，但爲了增加新的功能，它允許新增新的device driver，file system等。使用可載入核心模組(Loadable Kernel Module，簡稱LKM)便是擴充核心功能的一個最簡便的方法。

每個模組至少都必須提供init_module()和cleanup_module()這兩個基本的函式，前者是用來初始化這個模組，在模組載入核心時會被呼叫。後者是用來結束這個模組的工作，在從核心移除模組時會被呼叫。

圖4是從“Linux Device Drivers”書中所引用的一張圖。如圖所示，系統管理者可使用“insmod”指令動態地將一個模組載入核心，此“insmod”指令會呼叫模組中的“init_module( )”函式，作初始化的動作，並且向核心註冊此模組所提供的功能函式。如此一來，此模組所提供的功能函式便可以爲核心所用。當模組提供的功能不再需要時，系統管理者可使用“rmmod”指令動態地將模組從系統中移除。

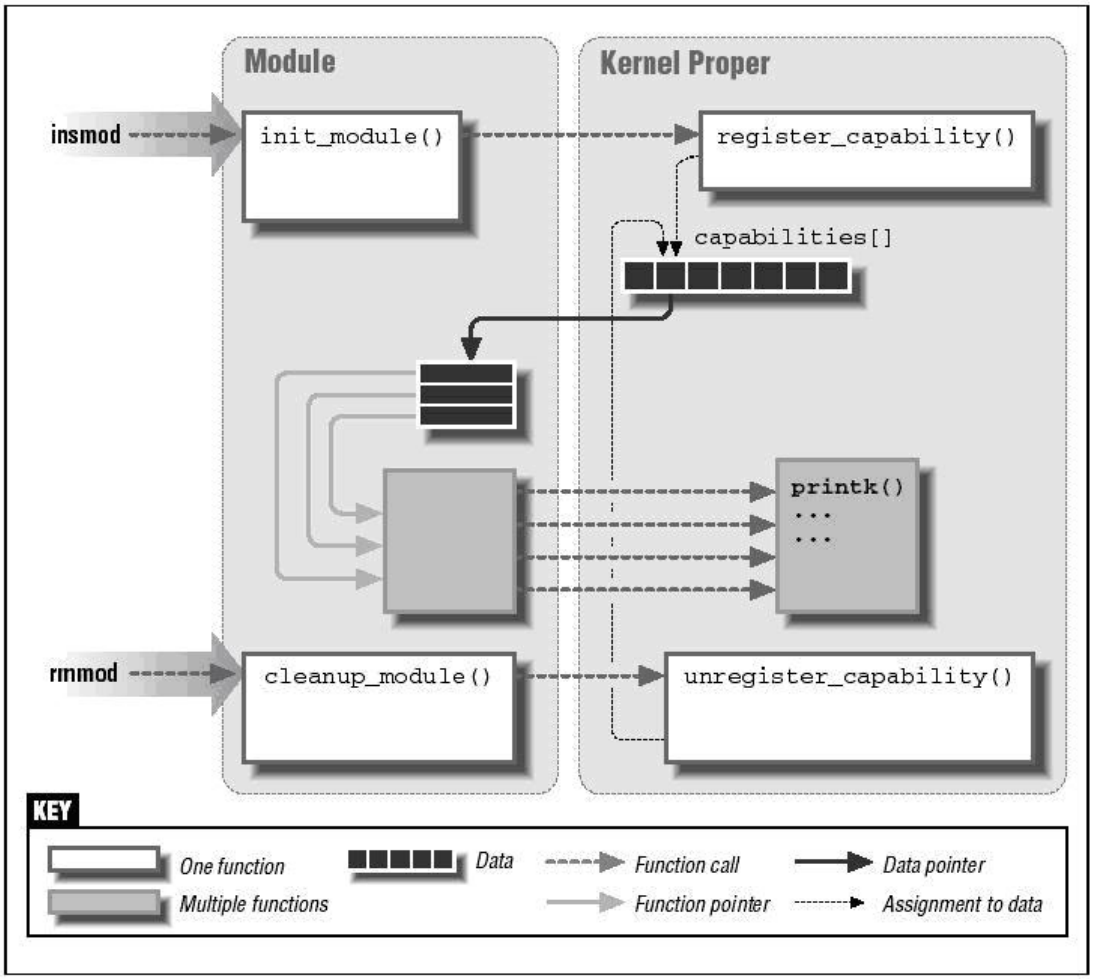


圖 4 核心載入模組

2.3.2 核心層之系統呼叫攔截

當一個程式想要存取系統資源時，它會向作業系統發出一個系統呼叫，此時系統呼叫的號碼會存入暫存器中，然後便由作業系統取得程式的控制權。當作業系統完成此次系統呼叫的工作後，程式的控制權便再回到要求存取資源的程式，如圖5所示，系統呼叫表(system call table)中存著指向處理該系統呼叫函式的指標。

正常狀況下，系統呼叫的處理流程如圖5的藍色路徑，作業系統會根據系統呼叫表中對應的函式指標，呼叫處理該系統呼叫的函式，完成後直接由處理系統呼叫的函式傳回結果給呼叫系統呼叫的程式。

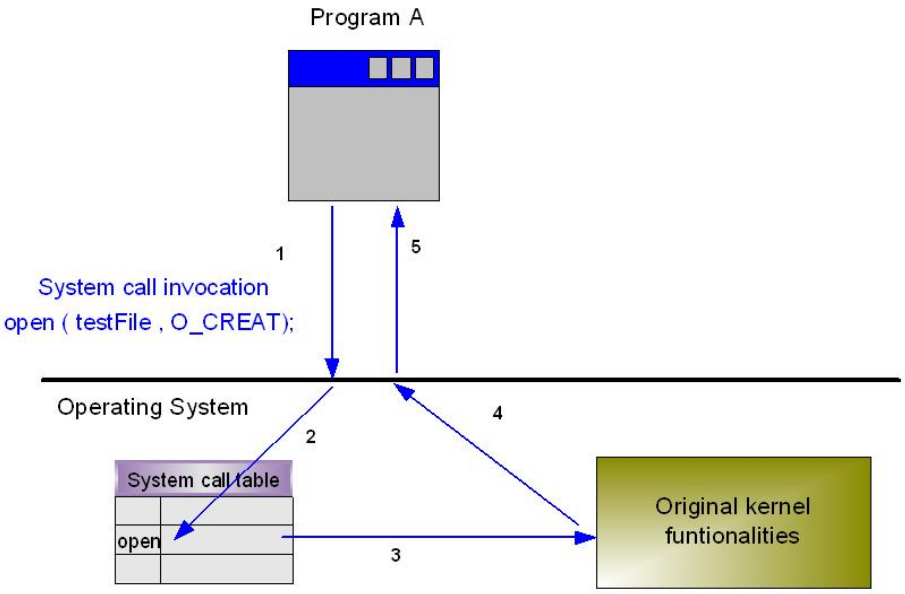


圖 5 正常系統呼叫處理流程

若系統呼叫表被修改，系統呼叫的處理便由原本直接執行核心的函式，改爲指到其他攔截者函式，如圖6中的橙色路徑。

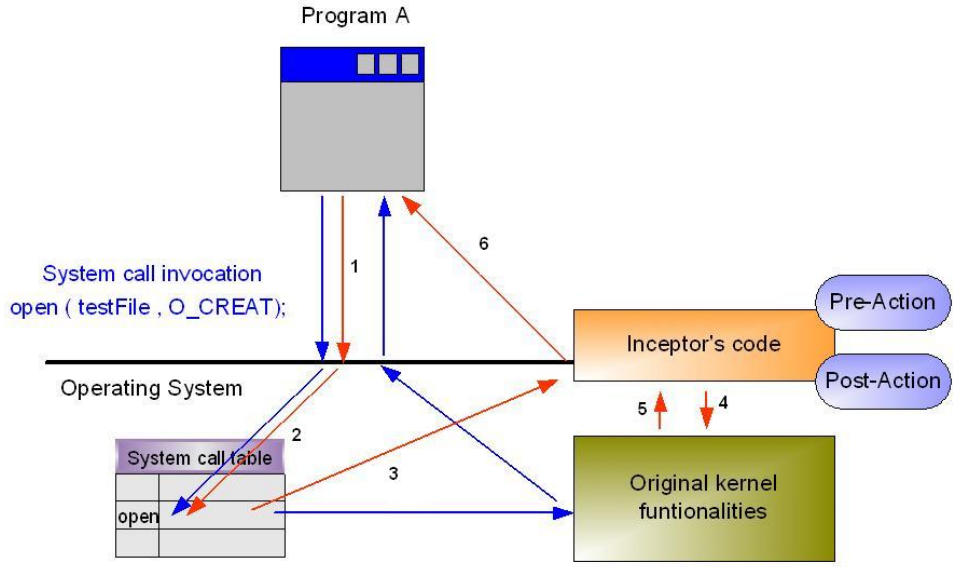


圖 6 攔截系統呼叫下的處理流程

通常在攔截系統呼叫時，最後都會呼叫原本核心中處理系統呼叫的函式，以保持一致性。攔截系統呼叫可由攔截者函式做一些動作，例如檢查參數等，最後再呼叫原本核心中的函式來處理此次系統呼叫。

在核心層攔截系統呼叫，利用LKM和device driver的技術，我們可以在LKM

中直接修改system call table的內容，將每個指到原本核心中處理系統呼叫函式的指標，改爲指向我們實作在LKM中的攔截函式。圖7是一個簡單的核心層系統呼叫攔截的例子。這個例子中實作了一個假的exit函式來取代原本的exit函式，然後在模組的init_module()函式裡，將系統呼叫表中指向原本exit函式的指標，改爲指向假的exit，並且在模組的cleanup_module()函式裡，把system call table復原，即把指標指回原本的exit函式。如此當這個模組被載入時，任何程式呼叫exit這個系統呼叫，都會被導到假的exit函式，而當這個模組被移除時，便會將system call table復原，使得exit這個系統呼叫會導到原本的exit函式。

```
1  extern void *sys_call_table[];
2
3
4  asmlinkage int (*original_sys_exit)(int);
5
6  asmlinkage int fake_exit_function(int error_code)
7  {
8          printk("HEY! sys_exit called with error_code= %d \n",error_code);
9
10         return original_sys_exit(error_code);
11 }
12
13 int init_module()
14 {
15         original_sys_exit=sys_call_table[__NR_exit];
16
17         sys_call_table[__NR_exit]=fake_exit_function;
18 }
19
20 void cleanup_module()
21 {
22         sys_call_table[__NR_exit]=original_sys_exit;
23 }
```

圖 7 核心層系統呼叫攔截範例

2.4 人體免疫系統(Human Immune Systems)

人體隨時會接觸到外來的物質，例如水，空氣等，這些外來的物質中很可能藏有危害人體的入侵者，例如細菌，濾過性病毒，甚至寄生蟲。人體中有免疫系統，負責偵測並消滅這些外來入侵者，肩負著保衛人體健康的重責大任，在本節中，我們將簡介人體免疫系統的運作情形，並且從中獲得設計入侵防禦系統的啓發。

人體免疫系統的主角是淋巴細胞(lymphocyte)，或稱淋巴球,、抗體(antibody)淋巴細胞的主要工作是分辨敵我，它可以分辨哪些細胞是人體本身的細胞，哪些是外來的、對人體可能有害的物質，這些可能有害的物質稱爲抗原(antigen)。一

個淋巴細胞的產生，是從基因庫裡面隨機選取一些基因片段來組成的，因此既然是隨機選取來組成，這樣產生的淋巴細胞便可能會是有問題的，可能無法達成分辨敵我的任務，所以必須經過一定的機制來檢驗這些細胞，如圖8所示。

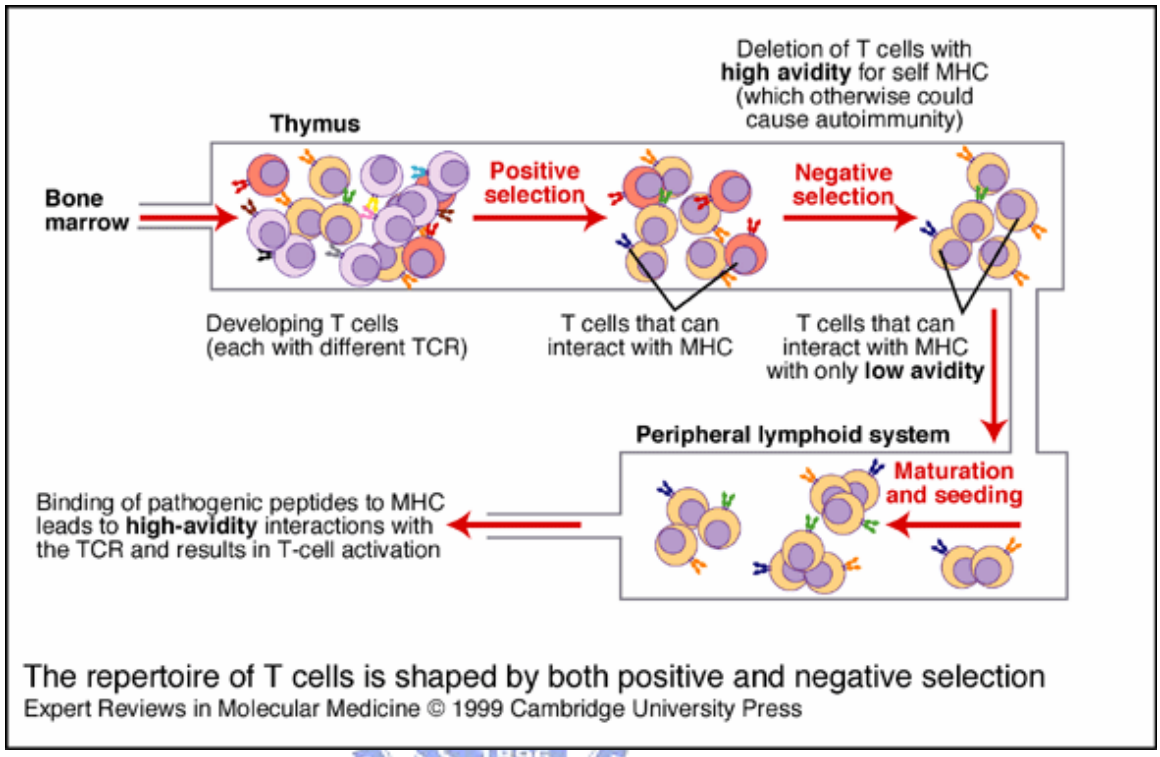


圖 8 淋巴細胞(T 細胞)的生成

一個新產生的淋巴細胞必須通過正向選擇(positive selection)和負向選擇(negative selection)兩項測試才能成爲成熟的淋巴細胞。正向選擇將會和其他自己細胞互相合作的細胞選擇保留下來，以方便以後的T-細胞能透過主要組織相容性原限制性(MHC-restriction)彼此合作。負向選擇就是讓這個淋巴細胞盡量去接觸人體的正常細胞，把會對自己抗原起反應(self-reactive)的細胞去掉，以避免把正常的細胞誤判成敵人，自己打自己，產生自体免疫反應(autoimmunity)。

若在一段時間內都沒有發生任何一次誤判，那麼就是一個成熟的淋巴細胞，具有分辨敵我的能力，便會放到血液裡面去面對可能進來的外來物質。

當成熟的淋巴細胞進入血液以後，它就會開始去找看看是否有它認識的外來物質(即會與之起反應的抗原)，如果一段時間內它偵測到足夠數量的外來物質，便會被活化(activated)，被活化以後的淋巴細胞就會開始去殺死這些它認爲是有害的細胞。一段時間內如果淋巴細胞沒有被活化，就表示這個淋巴細胞目前可能

沒什麼作用，便會死亡，由新產生的淋巴細胞代替。

被活化以後的淋巴細胞因爲要開始大量殺敵的任務，將進入clone selection的狀態，開始複製自己，這些複製出來的淋巴細胞叫做memory cell，會存在血液裡面比較長的時間去殺死那些有害細胞。

以上是整個人體免疫系統的機制，其運作方式與目的和我們設計入侵防禦系統的出發點相同，都是要偵測出入侵者並防止它們對人體或電腦系統造成傷害。

第三章、相關研究

本章節將介紹與本論文相關的一些研究。首先，介紹與本論文相關的入侵偵測系統，分別是近年來被廣泛討論的以攔截系統呼叫爲基礎之入侵偵測系統，以及使用狀態轉換觀念的入侵偵測系統。人體免疫系統防止病菌入侵人體，而入侵偵測防禦系統防止駭客入侵電腦系統，早在1996年便有研究開始將人體免疫系統的概念應用在入侵偵測防禦上，3.2將介紹以此概念實作的入侵偵測系統。近年來，不少研究指出現有IDS系統的缺點與漏洞，即使安裝了IDS仍有遭入侵的疑慮，3.3中，我們將介紹現有的主機型入侵偵測系統上可能遭受的攻擊，與規避其偵測的方法。

3.1 相關入侵偵測技術(Related IDS Techniques)

第二章中我們簡介過許多入侵偵測的分析方法，在本節中，將詳細介紹與本論文的實作直接相關的兩種入侵偵測技術。

3.1.1 N-gram 技術

一個惡意的破壞程式要對系統造成傷害，必須執行某些特權的動作來存取系統資源，而在現今的作業系統中，要存取系統資源唯一的方法是透過作業系統所提供的系統呼叫。因此，以檢查監督每個程式的系統呼叫來控管安全性是非常可行的方法，許多入侵偵測系統也朝著這個方向來設計，把系統呼叫作爲資料特徵來分析。

N-gram技術是由Stephanie Forrest等人最早在1996年提出[18]，並在1998年提出改進[17]。此技術以分析系統呼叫來設計入侵偵測系統，他們在2000年時提出以此技術爲基礎的pH(process homeostasis) IDS[1]。這個方法屬於anomaly IDS，觀察程式執行時呼叫的系統呼叫所形成之系統呼叫順序(system call sequence)，

即根據程式執行了哪些系統呼叫，加以分析後判斷它的合法性。它爲每一個特權程式(privileged processes)建一套正常行爲資料庫(self database)，這個資料庫是根據每個程式在特定的硬體架構、軟體版本和設定，在正常的使用狀況下建立出來的。表1 是一個建立正常行爲資料庫的例子，由以下的系統呼叫順序所建立出來：open，read，mmap，mmap，open，getrlimit，mmap，close

表 1 N-gram 建立正常行爲資料庫第一回合

call	Position 1	Position 2	Position 3
open	read	mmap	mmap
read	mmap	mmap	
mmap	mmap		

N-gram技術中的N代表觀察連續N個系統呼叫的關係，此例中N=3，表示觀察每個系統呼叫後跟著的三個系統呼叫是否正常，表1 爲建立正常行爲資料庫的第一個回合，window size=k+1=4，故觀察到第四個系統呼叫mmap，呈現的是前四個系統呼叫的關係，open後面跟的依序爲read，mmap，mmap，而read後面則依序爲mmap，mmap。此第一回合結束後，向右移動一個位置，第二回合觀察的便是第二到第五個系統呼叫，以此類推，整個系統呼叫順序共經六個回合，結果如表2所示。

表 2 N-gram 建立正常行爲資料庫

call	Position 1	Position 2	Position 3
open	read	mmap	mmap
	getrlimit		close
read	mmap	mmap	open
mmap	mmap	open	getrlimit
	open	getrlimit	mmap
	close		
getrlimit	mmap	close	
close			

將正常行爲資料庫建立起來後，即可用來和執行時的系統呼叫進行比對，因爲在此只看呼叫之系統呼叫的順序，不看所傳的參數，所以對監督程式來說，要做的工作只有攔截系統呼叫、與正常資料庫比對。若有一個新執行的系統呼叫順序：

open，read，mmap，open，open，getrlimit，mmap，close

與表2的正常行爲資料庫比對，可發現有四個不吻合的地方：

(1) 表2 open在position 3的位置並沒有open。

(2) 表2 read在position 2的位置並沒有open。

(3) 表2 open在position 1的位置並沒有open。

(4) 表2 open在position 2的位置並沒有getrlimit。

此種N-gram的比對方式，因爲只需比對系統呼叫順序，優點是簡單有效率，而偵測率和window size N有很大的關係，window size越大，對正常行爲的要求就越嚴謹，因爲要很長一串都完全吻合才算正常，如此可以提高偵測的效果，但相對的也會提高誤判率。

然而此方法只有比對系統呼叫順序，並沒有檢查系統呼叫所帶的參數，在偵測攻擊上稍嫌不足，有些攻擊者會利用合法的系統呼叫卻帶不合法的參數來進行攻擊。另外，若從它的正常行爲資料庫(如表2)中選一些不影響攻擊效果的系統呼叫塞進攻擊樣式裡，只要讓它超過window size，造成攻擊樣式被分散開來，它便無法偵測出攻擊了，這也是此種偵測方式的缺點。

3.1.2 以狀態轉換為基礎之技術 (State-Transition-Based Technique)

有限狀態機(Finite state machine，FSM)的運作是用一群各自不同的狀態以及描述一些觸發事件所組成的狀態轉換，各個狀態間皆有某種關係，且這些關係就是觸發狀態轉移的事件。我們可以將入侵行爲看成一連串的狀態轉換，以有限狀態機來描述一個攻擊行爲，如圖9是一個以狀態轉換來描述病毒感染的例子。本小節中，我們將介紹以狀態轉換爲基礎的入侵偵測技術。

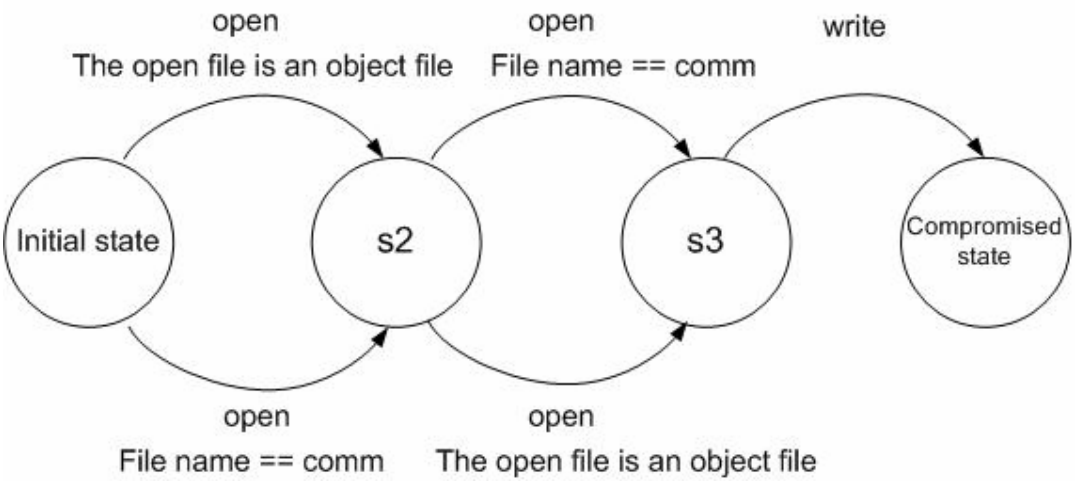


圖 9 以 FSM 描述攻擊行爲--病毒的感染

使用FSM的分析方法中比較著名的系統有由Reliable Software Group此單位所研究的STAT(State Transition Analysis Tool)[14][20]，有許多研究也是以STAT爲基礎，如STBIPW[24]。STAT使用圖形化的狀態轉換表示法，能夠比Rule-based方法更爲直覺地描述複雜的入侵行爲。在及時的環境中，STAT可利用已對一使用者監聽紀錄下的資訊來追蹤該使用者的行爲，並且判斷使用者目前的動作是否會對系統安全造成威脅。

STAT的架構與傳統的IDS架構非常類似(如圖10所示)，監聽資料前置處理器(audit record preprocessor)將原始的監聽資料重整爲系統可處理的格式。資訊庫(knowledge base)包含了規則庫(rule base)和參數庫(fact base)兩部份。規則用來建立使用者行爲和狀態轉換的關係，參數則是系統特定的資訊。介面引擎(inference engine)讀取所有的規則與監聽資料來判斷是否有入侵行爲。若有可能的入侵行爲符合規則，介面引擎便會對決策引擎(decision engine)發出警報，由決策引擎決定如何處理。

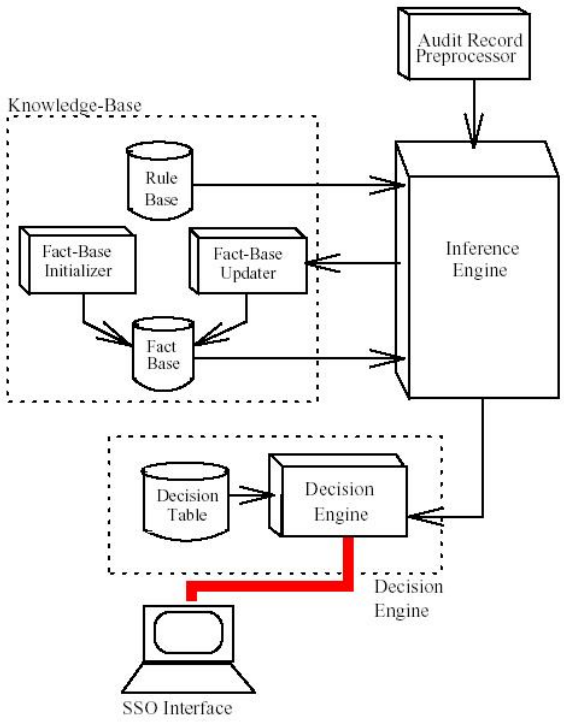


圖 10 STAT 架構

STAT能夠進行on-line和off-line的入侵分析。在off-line模式中，STAT分析已儲存的監聽資料來追蹤有嫌疑的非法行爲。在on-line模式中，每個入侵規則都會被轉換爲一個入侵樣板外掛模組(plugin)，動態地載入STAT的核心中。當蒐集到監聽資料時，會立刻送到這些外掛模組，即時地進行入侵偵測分析。

圖11將STAT當成一個元件，與其它入侵偵測元件整合成一個大型的入侵偵測系統，結合了多種入侵偵測技術將使得偵測能力更爲強大。

STAT將攻擊行爲以狀態轉換來表示，優點是攻擊描述簡單且直覺，讓使用者可以更容易的用狀態轉換圖來取代撰寫文字式的規則，省去學習描述攻擊行爲語言的時間，另外也因其簡單的特性，偵測攻擊時只需看下一個轉換狀態的條件，非常有效率。但STAT也受到FSM先天上的限制，在強度上稍嫌薄弱，例如沒有記憶功能，可能讓攻擊者利用繞路來達成攻擊，在4.2中我們會討論此種攻擊與防治的方法。

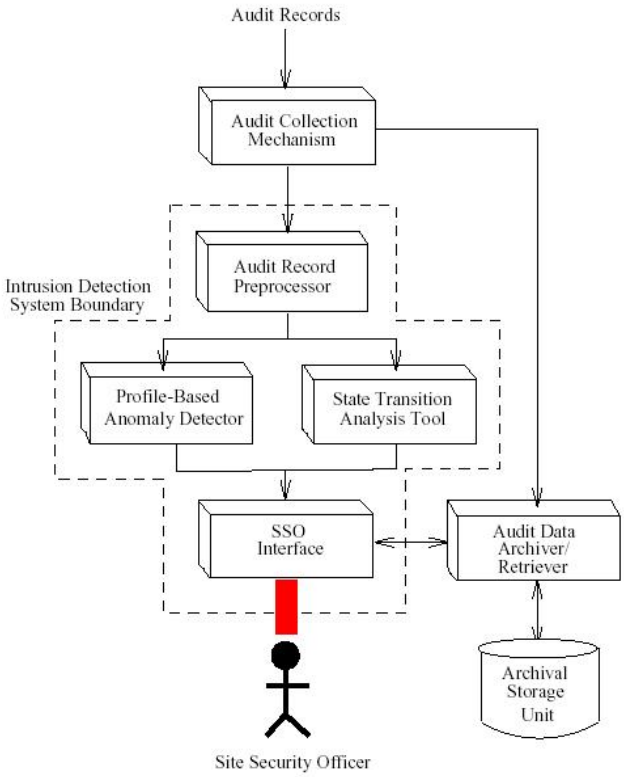


圖 11 入侵偵測元件組織圖

3.2 基於人體免疫系統之入侵偵測系統 (Immunity-Based IDS Systems)

人體免疫系統是保衛人體健康的一大功臣，它有效地阻止細菌、病源體和寄生蟲等外來的威脅侵入人體，而它有效率的運作機制也引起了包含電腦系統在內的各個學術研究領域的高度興趣。在資訊安全的領域，許多研究也紛紛提出以人體免疫系統爲基礎來設計的入侵偵測系統[28][29][30]。本節中，將介紹兩個以人體免疫系統爲基礎之入侵偵測系統。

3.2.1 基於人體免疫系統之網路型入侵偵測系統

Zhang Yanchao等人在2001年提出了基於人體免疫系統之網路型入侵偵測系統[28]，我們在第二章中已經介紹過人體免疫系統的運作機制，而這個網路型入侵偵測系統的運作機制與人體免疫系統十分類似。

一個網路連線可以經過一些規則，例如根據IP address，port number等等其他資訊，把它對應到一個長度爲L的位元串列(bit string)，而此系統中所謂的探測器(detector)，也是表示成一個長度爲L的位元串列。探測器擔任辨識敵我的角色，相當於人體免疫系統中淋巴細胞的功能。當一個網路連線所對應的位元串列和某個探測器吻合時，表示這個網路連線可能是一個攻擊的行爲。

圖12是實體機器分布的架構圖，主系統(primary system)負責產生探測器，然後把這些探測器散布到各個偵測系統(detector agent)。

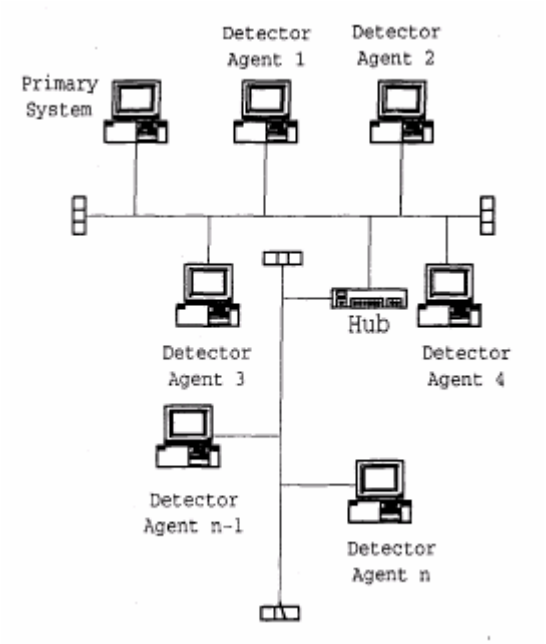


圖 12 基於人體免疫系統 IDS 之實體架構

圖13是系統的執行流程圖，上面的部分是在主系統中執行，包括隨機產生探測器，負向選擇(negative selection)與人體免疫系統中淋巴細胞的負向選擇機制十分類似，即讓隨機產生出來的探測器和一些正常的網路連線做比對，如果探測器會和這些正常的網路連線吻合，表示這個探測器無法使用，因爲它會將正常的行爲誤判成攻擊。通過負向選擇測試的探測器即可用來辨識非正常的網路連線，它們會組成一個個的群組，分散到各個偵測系統。

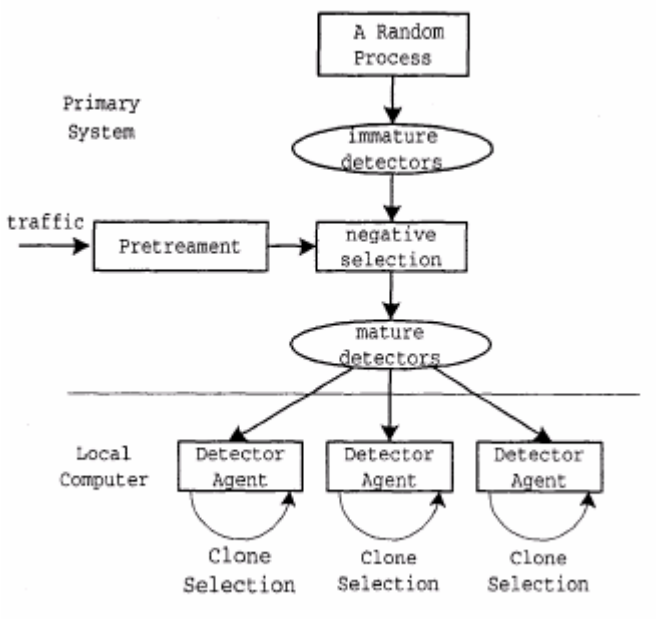


圖 13 基於人體免疫系統 IDS 之執行流程

圖14是一個探測器的生命週期，從產生出來到被消滅的整個過程。探測器通過負向選擇之後，就會進入開始偵測的階段，若在一定的時間內達到一定數量的吻合，探測器就會被活化，此時會發出一個警報給系統的安全管理者，決定要不要防堵這種攻擊。決定防堵的話就會把這個探測器複製成很多個記憶探測器(memory detector)，分散到各個偵測系統。

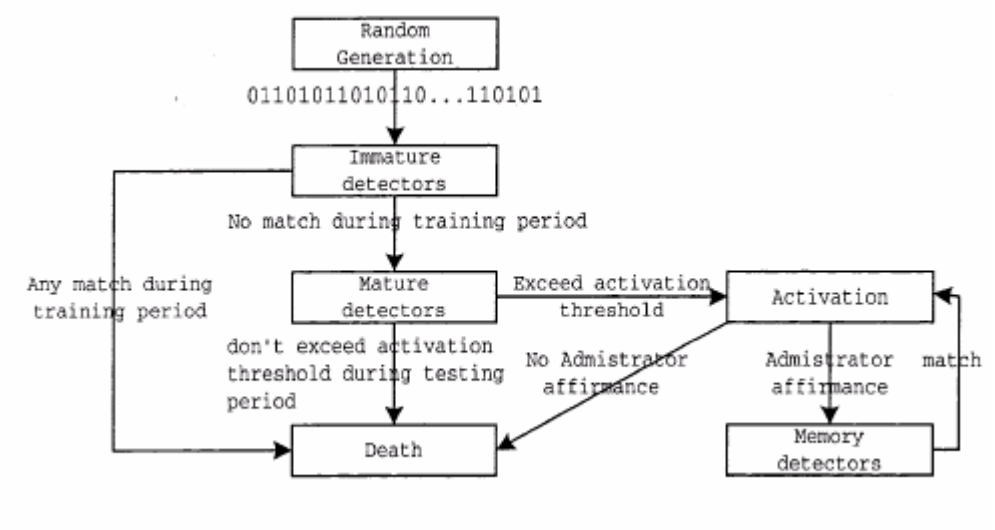


圖 14 探測器的生命週期

爲了減低誤判的機率，此系統使用兩個機制，第一個是活化門檻(activation threshold)，即探測器被活化然後發出警報的門檻。每個探測器都會設一個門檻，

在一定的時間內這個探測器必須要吻合很多次同樣的攻擊樣式才能被活化，以降低發生一兩次吻合就誤判的情形。另外有時候同一種攻擊可能從不同地方過來，單靠一個探測器可能偵測不到，所以可以在每個偵測系統上設一個門檻，當同屬於這個偵測系統的任何一個探測器發生吻合時，就將計數值加一，加到超過門檻時表示遭受攻擊，探測器便可以被活化並發出警報。

此網路型入侵偵測系統將攻擊樣式隨機產生出來再用基因演算法做篩選，因此不需要使用者自己定義攻擊樣板，基本上屬於anomaly的IDS。然而它的效率上有些遭受質疑，因爲要隨機產生出真正爲駭客所用、會造成系統傷害的攻擊樣式機率並不是很高，加上產生出來的攻擊樣式若過一段時間沒偵測到攻擊變會被刪除，因此必須在攻擊行爲發生時，剛好有隨機產生出可以偵測它的攻擊樣板才可以偵測出來，偵測效率可能不高。

3.2.2 IGSTAM

Immune Genetic State Transition Analysis Model(IGSTAM) [30]是Zhou-Jun Xu等人在2003年所提出，結合人體免疫系統概念與STAT分析方法的主機型入侵偵測系統。此系統中定義的FSM狀態集合(state space) S爲作業系統中代表一個資源的各項參數：S={ file_name，full_path，owner，member，eid，gid，permitted，located，same_user，same_pid，shell_script }，FSM狀態轉換函式集合(system signature action space) A代表一個使用者可能執行的動作：A={ read，wirte，create，execute，exit，delete，modify_ower，modify_perm，hardlink，rename }。表3爲一個入侵流程的例子，用FSM來表示如圖15。

表 3 mail utility 入侵流程

Step	Command	Comment
1	%cp /bin/csh /usr/spool/mail/root	assumes no root mail file
2	%chmod 4755 /usr/spool/mail/root	make setuid file
3	%touch X	create empty file
4	%mail root < X	mail root empty file
5	%/usr/spool/mail/root	execute setuid-to-root shell
6	root%	

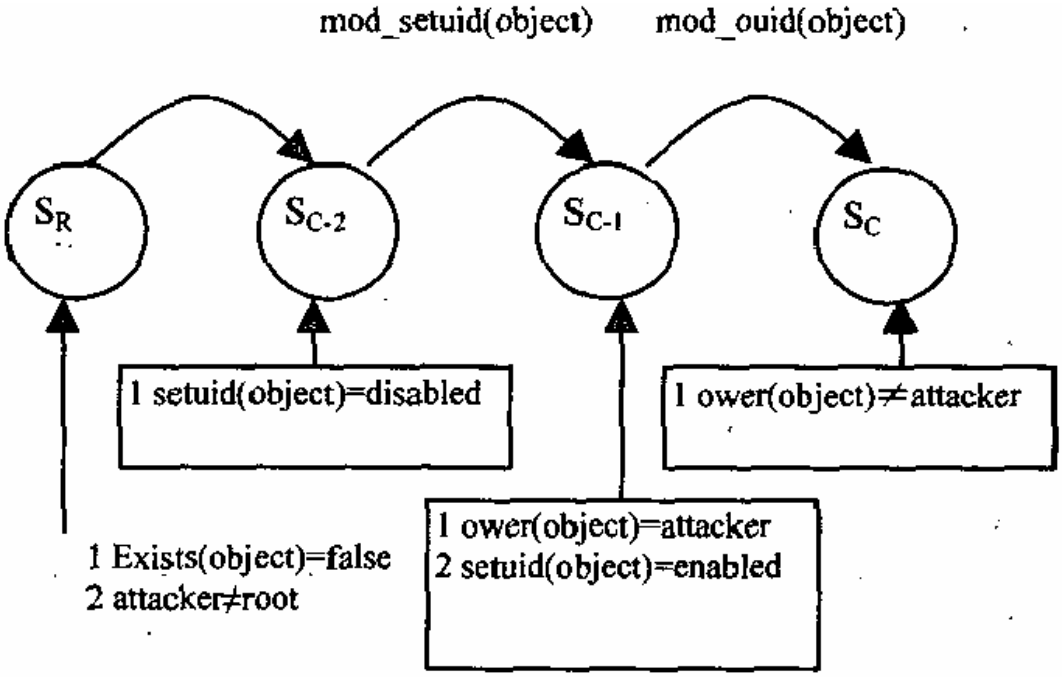


圖 15 mail utility 入侵流程之狀態轉換表示圖

IGSTAM中將入侵行爲表示成狀態轉換串列(state-action chain)，如：SA=(S1,A1,S2,A2,…Sn)，系統中所謂的疫苗(Vaccine)相當於3.2.1中的探測器，代表可能的入侵行爲，也是表示成如上的狀態轉換串列。

圖16是IGSTAM的流程圖。疫苗的產生使用人體免疫系統的方法，先隨機產生一個狀態轉換串列 Vac= (S1,A1,S2,…An-1,Sn)。接著產生的疫苗要經過Non-self selection，相當於3.2.1的負向選擇，即接觸一些正常行爲的狀態轉換串列，進行比對，看是否會跟這些正常行爲的狀態轉換串列吻合。若經過一段時間的比對皆無吻合，便成爲成熟的疫苗。

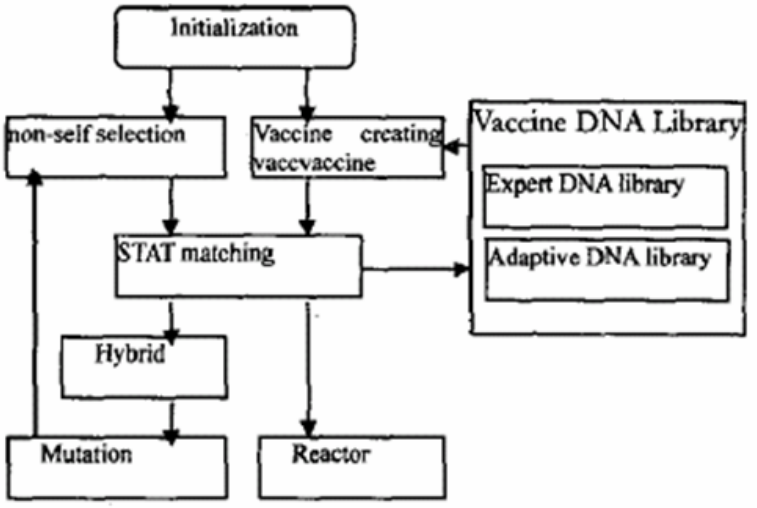


圖 16 IGSTAM 流程圖

疫苗成熟以後就把它放到適應型DNA庫(adaptive DNA library)，所謂的DNA庫是一些狀態轉換串列的集合，DNA庫分爲兩種，一種是專家型DNA庫(expert DNA library)，裡面存已既定是攻擊行爲的疫苗，即代表攻擊行爲的狀態轉換串列。另一種是適應型DNA庫，裡面存由免疫機制產生出來的新疫苗。

在適應型DNA庫裡的疫苗可以表示成 VACCINE={ Vac , A }，Vac是狀態轉換串列，後面的A是一個參數，當這個疫苗偵測到攻擊時，A值會增加，相當於3.2.1中的活化機制，一段時間以後如果這個A的值一直沒增加，此疫苗便會從這個適應型DNA庫裡被刪除，讓DNA庫裡的疫苗保持最新最符合目前的需要。

混合機制(Hybrid)是一種基因演算法，例如當有兩個疫苗Vac1 和 Vac2成功地偵測到攻擊時，就把這兩個狀態轉換串列隨機組合成一個新的Vac3，這是爲了較容易產生可以偵測相似攻擊的疫苗。

突變機制(Mutation)也是一種基因演算法，經過混合機制的疫苗有一定的機率會再改變幾個值，然後再回到負向選擇做測試。

IGSTAM的優點是導入更多基因演算法，若是將已知的攻擊行爲導入，有較大的機會可以偵測到變種的攻擊。然而隨機產生的疫苗在偵測效率上與3.2.1介紹的系統有著同樣的問題，即隨機產生的新疫苗在偵測新型攻擊上的效率可能不高，因此我們提出的系統不採取隨機產生攻擊樣式的方式，而只取其中負選擇作用的方法來檢驗使用者定義的攻擊樣板。

3.3 主機型入侵偵測系統上之攻擊

雖然入侵偵測技術行之已久，許多文獻對IDS也不斷提出改進的方法[6][9][26]，期望能夠完全地防堵攻擊，但駭客的攻擊方式亦不斷翻新，加上IDS作法公開地被廣泛討論[2][27]，若一個IDS的防禦方式被知道的話，駭客很可能找出一些規避的方法來逃過偵測而進行攻擊，也有許多研究紛紛提出破解IDS防禦的方法[4][13]。本節中，將討論兩種規避IDS偵測而進行攻擊的方法。

3.3.1 插入 no-op 系統呼叫之攻擊

以定義攻擊樣板來偵測入侵行爲的IDS，必須偵測到某項吻合的樣式才可判斷出入侵行爲，因此若在一段攻擊樣式中，插入很多沒有作用(no-operation)而且

不影響攻擊效果的元素，便可以混淆IDS對攻擊樣式的辨識，例如在3.1.1中介紹的N-gram技術，必須要連續幾個系統呼叫所形成的系統呼叫順序爲異常，才會被判斷爲異常的攻擊行爲。但若把攻擊行爲的系統呼叫順序打散，如圖17所示，底線的部份連續出現的話會被IDS偵測出來，這時使用IDS的正常行爲資料庫中的系統呼叫，穿插在攻擊樣式中將其打散，如圖17未畫底線的部份，則插入的no-op系統呼叫將使得IDS無法辨識出原本的攻擊樣式，而無法偵測出入侵行爲。

進行no-op攻擊的基本假設，是攻擊者能夠知道哪些屬於正常的系統呼叫，而此假設成立的條件並不困難，因爲有非常多基本常用的系統呼叫可供利用，且若持續觀察IDS系統的行爲，也可推測出屬於正常行爲的系統呼叫，甚至可推測出IDS系統所使用的演算法。若在misuse的IDS系統中，則因不需要考慮到正常行爲資料庫，此種攻擊更容易成功，只需把原本的攻擊樣式插入一些no-op的系統呼叫即可。

將攻擊行爲隱藏在正常行爲中，稱爲擬態攻擊(Mimicry Attacks)[4]，此插入no-op的攻擊法也屬於擬態攻擊的一種。

```
read() write() close() munmap() sigprocmask() wait4()
sigprocmask() sigaction() alarm() time() stat() read()
alarm() sigprocmask() setreuid() fstat() getpid()
time() write() time() getpid() sigaction() socketcall()
sigaction() close() flock() getpid() lseek() read()
kill() lseek() flock() sigaction() alarm() time()
stat() write() open() fstat() mmap() read() open()
fstat() mmap() read() close() munmap() brk() fcntl()
setregid() open() fcntl() chroot() chdir() setreuid()
lstat() lstat() lstat() lstat() open() fcntl() fstat()
lseek() getdents() fcntl() fstat() lseek() getdents()
close() write() time() open() fstat() mmap() read()
close() munmap() brk() fcntl() setregid() open() fcntl()
chroot() chdir() setreuid() lstat() lstat() lstat()
lstat() open() fcntl() brk() fstat() lseek() getdents()
lseek() getdents() time() stat() write() time() open()
getpid() sigaction() socketcall() sigaction() umask()
sigaction() alarm() time() stat() read() alarm()
getrlimit() pipe() fork() fcntl() fstat() mmap() lseek()
close() brk() time() getpid() sigaction() socketcall()
sigaction() chdir() sigaction() sigaction() write()
munmap() munmap() munmap() exit()
```

圖 17 插入 no-op 系統呼叫之攻擊

3.3.2 繼承程式間之合作攻擊

如果攻擊者程式可以執行fork的話，一般的IDS通常會把fork出來的程式當作獨立的程式監控，也就是分別追蹤父程式和fork出來的子程式，但是若父程式和子程式合作進行攻擊的話，便不易偵測出來，如下例。

假設有一個攻擊樣式的系統呼叫順序爲：

open(“/etc/passwd”)

write(“/etc/passwd”)

close()

單一程式依序執行以上三個系統呼叫，會被IDS偵測爲入侵行爲，但程式可先執行fork，產生出新的一個子程式，然後由兩個程式合作進行攻擊，如圖18所示，由父程式執行open(“/etc/passwd”)，之後將所得到的檔案描述子(file descriptor)利用IPC(Interprocess Communication)的方式傳給子程式，最後由子程式執行write(“/etc/passwd”)、close()完成攻擊。IDS若把每個程式當成獨立程式監控，則因爲不論是父程式還是子程式都沒有完全吻合攻擊樣式，故無法偵測出此種程式間的合作攻擊。此合作攻擊法亦屬於擬態攻擊的一種，我們將在第四章中討論解決此問題的方法。

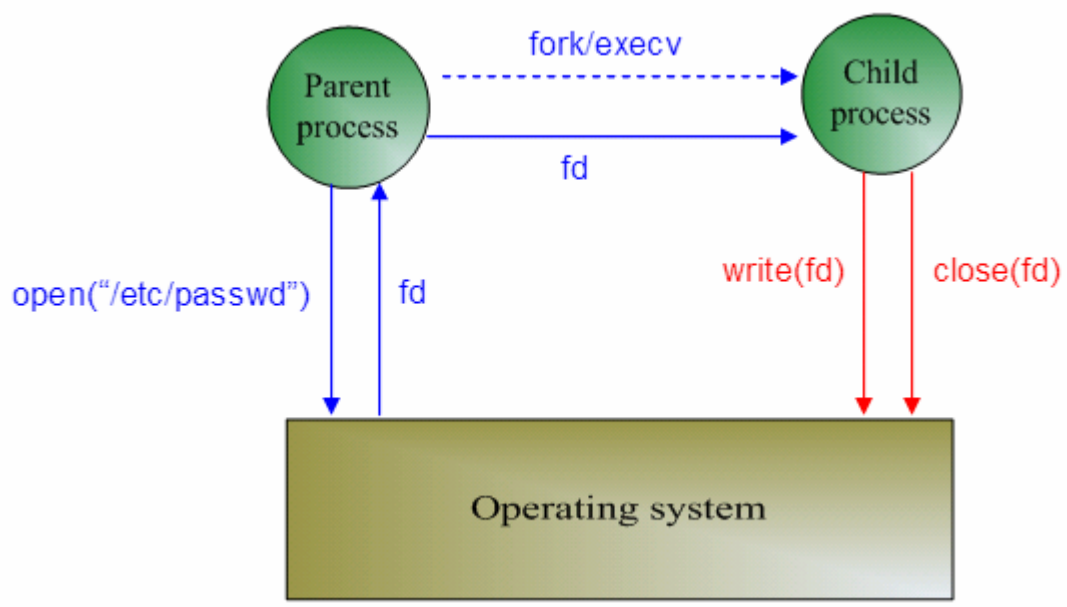


圖 18 繼承程式間之合作攻擊

第四章、系統需求與設計方法

本章節將介紹本論文所設計的系統之需求，以及設計方法。一直以來IDS系統面臨到的挑戰始終都是偵測率和誤判率的問題，IDS的設計者總是希望能夠設計出高偵測率與低誤判率的系統，在4.1中，將介紹從人體免疫系統得來的啓發，使用負選擇作用的觀念來預警不當之攻擊樣板，進而降低誤判率的方法。面對駭客日新月異的攻擊手法，IDS系統的設計也必須考量本身可能遭受的破解，在4.2中，我們將介紹防止插入no-op系統呼叫來規避偵測的攻擊，而4.3中我們將介紹防止另一種規避偵測的方法，即防止程式間合作進行攻擊，使本論文提出的IDS系統能夠更精準地偵測入侵行爲，並防止蓄意規避偵測。

4.1 不當攻擊樣板之檢測(Examine Improper Penetration Pattern)

誤判對入侵偵測系統來說是一個很嚴重的問題，過多的誤判對系統安全管理人員來說是沉重的負擔，同時也會浪費許多系統資源。誤判分爲兩種，一種是正向誤判(False positive)，即將正常的行爲誤判成攻擊行爲，另一種是負向誤判(False negative)，將攻擊行爲誤判爲正常行爲，意即沒有偵測出攻擊。較被廣爲討論研究的誤判屬於正向誤判，本論文亦以此爲研究重點。

無論是anomaly還是misuse的IDS都會有誤判的情形，anomaly IDS的誤判來自於正常行爲資料庫不夠健全，沒有將所有的正常行爲包含進去，因此未被蒐集到的正常行爲就會被誤判成攻擊。misuse IDS的誤判來自於攻擊樣式的定義模糊，使用者定義的攻擊樣式可能會包含到正常行爲因而造成誤判。本節中，我們將介紹使用基因演算法來減少誤判的方法。

4.1.1 建立正常行為資料庫(Normal Behavior

Database)

本論文提出的系統架構在偵測模式上屬於misuse IDS，讓使用者可以依照需求自行定義攻擊樣板，並以此樣板來偵測攻擊。在減少誤判的作法上，我們蒐集正常行爲，且用這些正常行爲來檢驗攻擊樣板，分爲兩個階段：訓練階段(Training phase)和測試階段(testing phase)，本小節介紹訓練階段。

訓練階段必須在保證沒有攻擊會發生的安全環境中，記錄下正常的系統呼叫順序，建立出正常行爲資料庫，此種建立正常行爲的模式類似anomaly IDS中常見的建立正常行爲資料庫，例如3.1.1中介紹的N-gram便是用此方法，但在anomaly IDS中，正常行爲資料庫是用來偵測攻擊，而我們的正常行爲資料庫則是用來檢驗使用者定義的攻擊樣板。

建立正常行爲資料庫時，因爲需要在安全的環境中，最好是將網路連線關閉，或是只允許特定的使用者帳號登入使用。雖然有些學術單位提供一些程式的執行記錄，但並不保證其中絕對沒有攻擊行爲。通常一個主機型入侵偵測系統會自己訓練正常行爲資料庫，以訓練出專屬於安裝此入侵偵測系統的正常行爲，同時也不受限於程式類型，可爲任何應用程式訓練出專屬的正常行爲資料庫。

4.1.2 負選擇作用(Negative Selection)之方法

在測試階段中，用的是人體免疫系統中負選擇作用的概念，如圖19，即讓使用者定義的攻擊樣板和我們在訓練階段建立出來的正常行爲資料庫做比對，來檢驗使用者定的攻擊樣板。此相當於人體免疫系統中，讓淋巴細胞盡量接觸身體內的正常細胞。

比對中如果有發生吻合，表示使用者定的攻擊樣板會把在正常行爲資料庫裡的所謂正常行爲當成是攻擊，這時候便要讓使用者決定，看是要檢查並修改攻擊樣板，或者就是不允許這個行爲發生，仍然要把它當作攻擊來阻止預防。若使用者的決定是後者，就必須把發生吻合的系統呼叫順序從正常行爲資料庫中刪除。透過這層檢驗，可以避免使用者定出不夠精準的、會產生許多誤判的攻擊樣板，降低因誤判造成的損失。

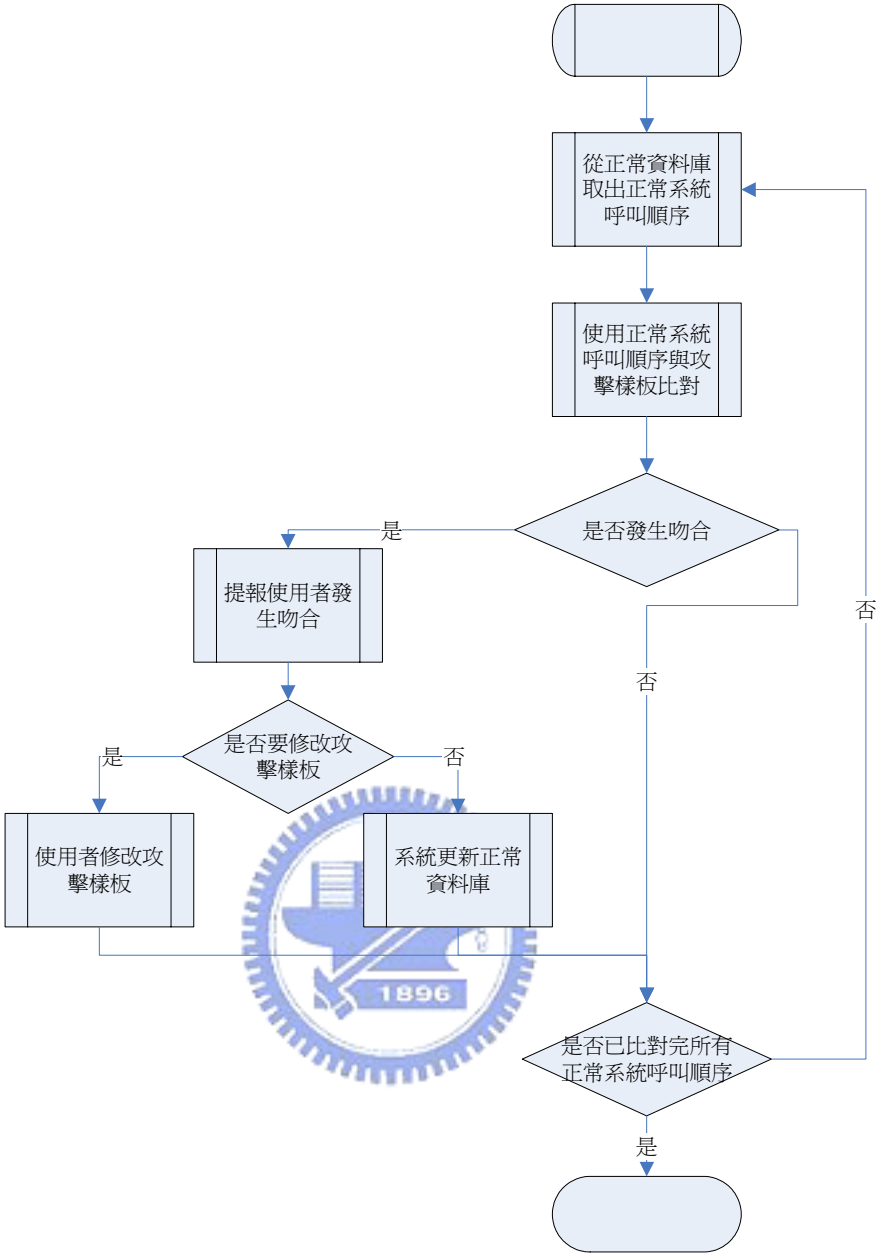


圖 19 負選擇作用之方法

4.2 防止插入 no-op 系統呼叫之攻擊

在 3.3.1 中我們介紹過，像 N-gram 那種使用樣式比對、吻合方式的作法，很容易用插入許多 no-op 的系統呼叫來規避偵測。然而在 STAT 中，只看關鍵事件(key event)來判定攻擊，即遇到關鍵的系統呼叫才會作狀態轉換的動作，插入無關的 no-op 系統呼叫並不會轉換狀態，看起來似乎不會有 no-op 攻擊的疑慮，但其實不然。例如，原本使用者定的攻擊樣式 FSM 如圖 20 所示，假設 a~f 皆爲

系統呼叫，若直接拿圖 20 來做偵測，因爲此 FSM 爲一個 DFA(Deterministic Finite Automaton)，遇到可轉換的條件時必定要轉換，因此像 ace 這樣的系統呼叫順序便無法偵測出來。其主要原因是 FSM 沒有記憶的功能，只能看目前的狀態，以及下一個狀態轉換的條件爲何，雖然這也是它有效率的地方。攻擊者若要避開對 ce 這個系統呼叫順序的偵測，可先執行 a，再執行 ce 即可，我們可以稱這樣的攻擊法爲繞路攻擊(evasion attack)，對這樣的攻擊來說，a 便算是一個 no-op 的系統呼叫，用它來擾亂 IDS 的偵測。

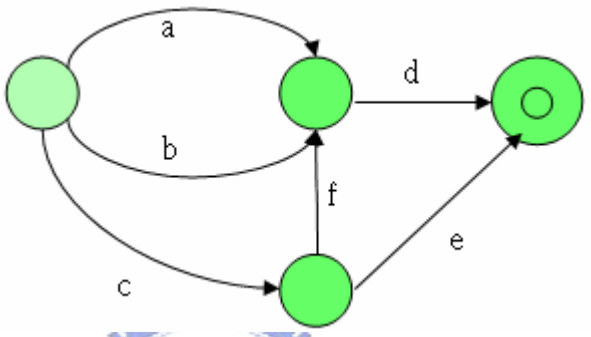


圖 20 原始 FSM 攻擊樣板

爲了解決這個問題，我們必須對原本的 FSM 作一些處理。首先，若使用者定的 FSM 比較複雜，我們會以到達最終狀態的路徑來考慮，先將它拆成幾個 sub-FSM，如圖 21。所謂的比較複雜，是指像圖 20 中，有兩條分叉路徑在終點前就交會的情形，較正式地說，即使用深度優先搜尋法(Depth First Search ， DFS) 來追蹤這個圖時，若過程中遇到重複走的邊，就必須把包含此重複走的邊之路徑獨立拆開出來。圖 22 爲檢查並將一個 FSM 拆爲多個 sub-FSM 之演算法。

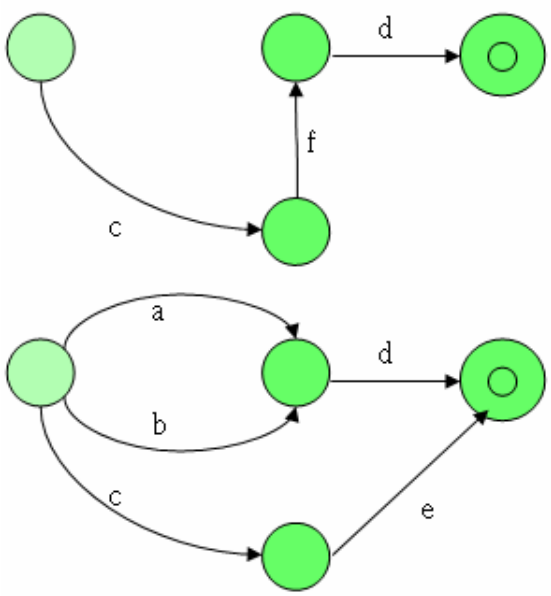


圖 21 sub-FSM 攻擊樣板

```
G:graph (FSM)
V[G]:nodes in G
U[G]:edges in G
P[G]:path from initial node to final node in G

DFS(G)
      for each node u  ∈  V[G]
          do   color[u] <--WHITE
               π[u] <--NIL
      for each node u  ∈  V[G]
          do   if   color[u] = WHITE
                    then DFS-VISIT(u)
DFS-VISIT(u)
     color[u] <--GRAY
     for each v  ∈  Adj[u]
          do   if   color[v]=WHITE
                    then  π[v] <--u
                             DFS-VISIT(v)
               if   color[v]=BLACK   and   v!=final   and   π[v] != u
                    then for each   p  ∈  P[G] that include (u,v)
                              do New a sub-FSM
                         discard (u,v) from G
     color[u] <--BLACK
     if   Adj[u] = NIL   and   u is not final
          then discard ( π[u],u) from G
               discard u from G
```

圖 22 sub-FSM 演算法

第二個步驟，是在每個 sub-FSA 的非終點狀態上，都加上一個回到前面分叉點的 ε 邊，成爲一個 ε -NFA(Nondeterministic Finite Automaton)，如圖 23。所謂的分叉點，是指有兩個以上的狀態轉換條件之狀態，圖 24 爲加上 ε 邊的演算法。加上回到分叉點的 ε 邊，表示不管現在處於哪個狀態，都有可能再從前面分叉的地方走另一條路徑來進行攻擊。

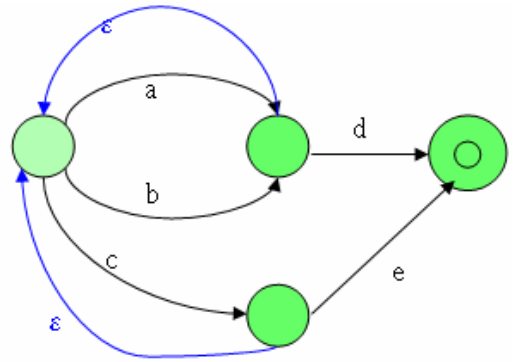


圖 23 ε -NFA

```
G:graph (FSM)
V[G]:nodes in G
U[G]:edges in G
D[G] : crotched nodes in G

DFS(G)
     D[G]=NULL
     for each node u ∈ V[G]
         do   color[u] <--WHITE
              π[u] <--NIL
     for each node u ∈ V[G]
         do   if   color[u] = WHITE
                  then DFS-VISIT(u)
DFS-VISIT(u)
    color[u] <--GRAY
    if   Num(Adj[u]) > 1
       then   Add   u   to D[G]
     for each v ∈ Adj[u]
         do   if   color[v]=WHITE   and   v!=final
                  then  π[v] <--u
                        for each d ∈ D[G]
                            do   Add (d,v) labeled ε
                        DFS-VISIT(v)
    color[u] <--BLACK
    if   u ∈ D[G]
         then discard u from D[G]
```

圖 24 加上 ε 邊演算法

最後利用正規語言(formal language)的方法，將 ε -NFA 再轉成 DFA，如圖 25 所示。須轉爲 DFA 是基於偵測時的效率考量，因爲 DFA 明確地定義遇到特定的條件會轉到特定的狀態，比定義轉換條件不明確的 NFA 有效率。如此轉出來的 DFA 便可防止 no-op (或繞路)的攻擊，例如原本在圖 20 中無法偵測的 ace、cad 這樣的系統呼叫順序，在圖 25 中在都可以偵測出來。故我們透過對使用者定的 FSM 的一些分析處理，可以確保 no-op 攻擊無法成功。

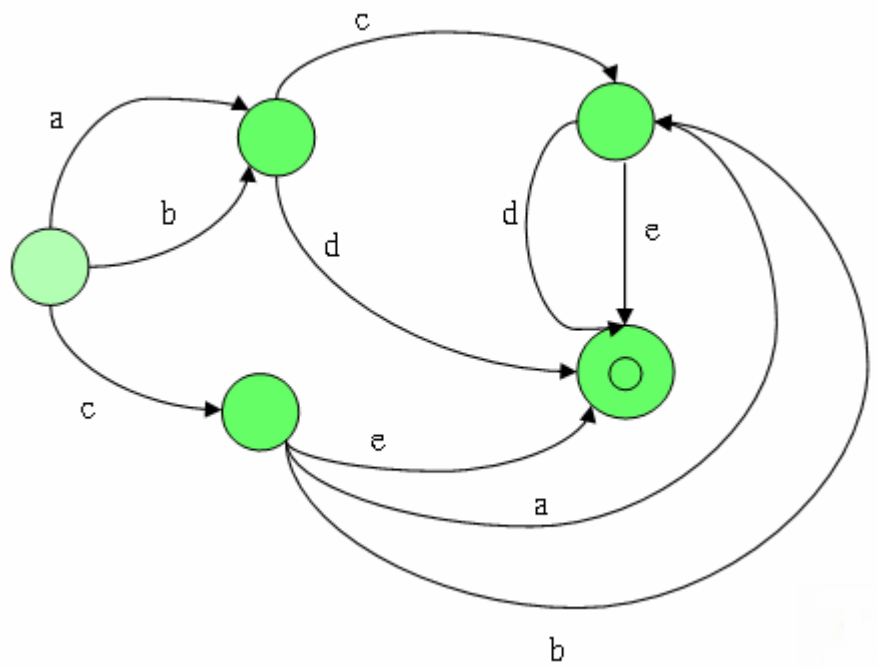


圖 25 ε -NFA 轉 DFA

4.3 防止繼承程式間之合作攻擊

防止合作攻擊的部份，我們假設合作攻擊的起點爲一個程式執行 fork 指令、準備產生一個子程式來進行合作，因此我們攔截 fork 系統呼叫做特別處理。當一個需要被監督的程式執行時，系統會產生此程式的 FSM 實體(instance)，隨著此程式執行系統呼叫，FSM 實體的狀態也會跟著轉換，用它來監督這個程式的執行。

若此程式執行 fork 系統呼叫，我們就會爲 fork 出來的子程式產生監督它的 FSM 實體，它的狀態會在初始狀態，之後隨著子程式執行系統呼叫，子程式的 FSM 實體的狀態也會跟著轉換。同時，也會產生一個家族 FSM 實體，狀態則與

執行 fork 系統呼叫之父程式的狀態一致，如圖 26 所示。

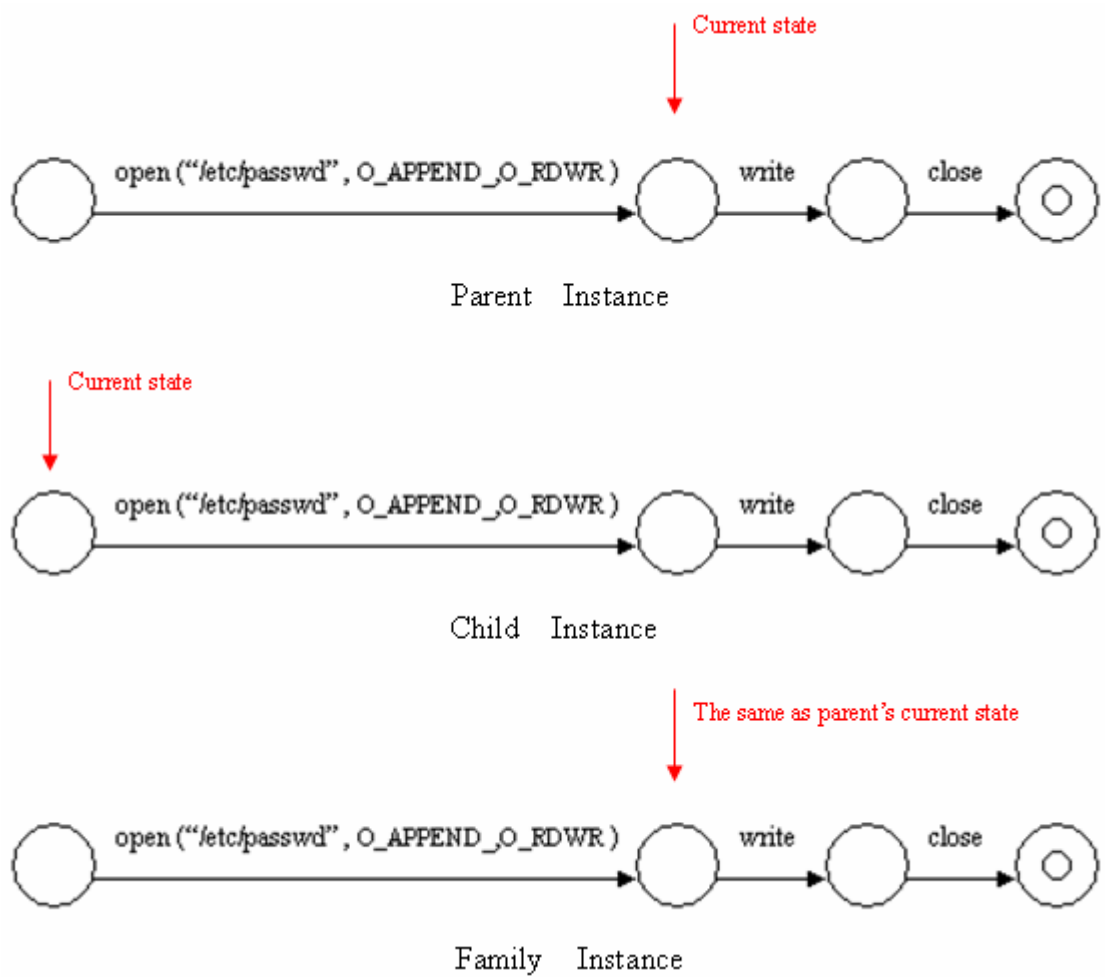


圖 26 使用家族 FSM 實體防止合作攻擊

父程式和子程式都有各自監督它們的 FSM 實體，防止它們獨立進行攻擊，而家族 FSM 實體，則是用來防止合作進行攻擊。不論是父程式或子程式，執行系統呼叫時，除了檢查更新自身的 FSM 實體外，也會同時檢查家族 FSM 實體是否需要更新，如圖 26，由父程式開啓 passwd 這個檔案後，不管是父程式還是子程式去寫入，家族 FSM 實體都會轉換到下一個狀態。因此透過建立這種監督家族的觀念，可以防止程式間的合作攻擊。

第五章、系統架構

本章節將介紹本論文所設計的AMA-IPS系統之架構，以及各個元件的簡介。一般以攔截系統呼叫的方式來監督程式執行，可實作在使用者層(user level)或核心層。實作在使用者層，優點是不需要更改作業系統核心，因此修改容易，擴充性較高。但因爲執行每個系統呼叫時，都必須先把控制權交給作業系統，再由作業系統將控制權給監督程式，之後控制權再回到作業系統執行系統呼叫，如此多了一來一回的環境切換(context switch)時間，在執行時間上的負擔很大。實作在核心層，優點是可省下環境切換時間，速度快很多，然而因爲修改到作業系統核心，要擴充功能時較爲不便。本篇論文提出的AMA-IPS系統採權衡的方式，將可靜態作的分析工作，在使用者層做完，且若要對攻擊行爲做更進一步的分析，也可繼續擴充功能。而及時的監督與偵測工作則放在核心層，降低執行時的時間負擔。5.1中，將簡介整個系統的架構與各元件之間的關係。5.2介紹使用者層各模組元件之功能。5.3介紹核心層各模組元件之功能。

5.1 系統架構概觀

我們的系統以STAT爲基礎，分爲使用者層元件和核心層元件兩大部分，兩部分元件的溝通使用device driver，整體系統架構如圖27。我們使用核心層軟體包覆隔離(Kernel-Level Wrapper)技術，將各核心層元件(component)實作成LKM(Loadable Kernel Module)，掛載於Linux系統的核心下。使用者層元件由使用者操作輸入後，將結果透過Wrapper driver輸出給核心層元件使用。

在訓練正常行爲階段，Normal Behavior Collector會紀錄正常使用狀況下的系統呼叫順序(normal system call sequence)，並將其儲存於正常行爲資料庫(Normal Behavior Database)中。

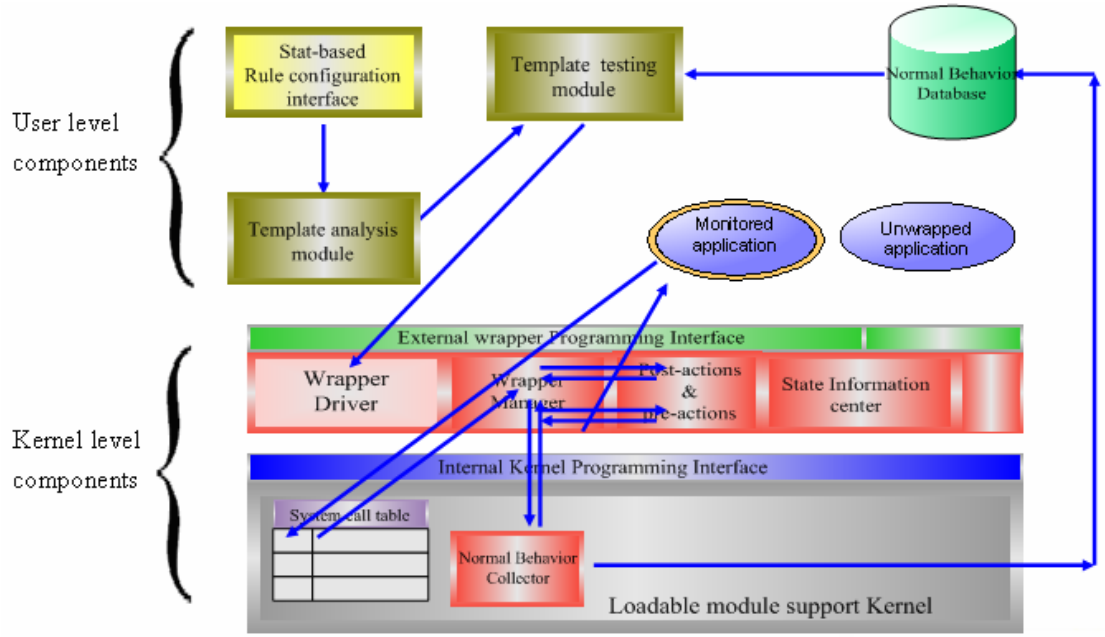


圖 27 系統架構圖

使用者在State-based Rule configuration interface制定入侵防禦規則，此模組有圖形化介面提供使用者定義攻擊樣式，制定完成後這個攻擊樣式我們稱爲攻擊樣板(template)，此攻擊樣板還不會直接用來偵測攻擊，而是先經過Template analysis module進行分析，轉爲可以抵抗no-op攻擊的DFA，之後再交由Template testing module做測試，檢驗使用者定的攻擊樣板是否會造成很多誤判情形。攻擊樣板通過檢驗後才可經由Wrapper Driver載入核心，這些攻擊樣板被載入後存在State Information center中。

核心層元件根據使用者定義的攻擊樣板來防堵入侵攻擊行爲。Wrapper Manager是系統中最重要的一個元件，負責產生攻擊FSM實體(Instance)、即時攔截系統呼叫、並檢驗此系統呼叫是否允許執行。Wrapper Manager會根據儲存在State Information center中的攻擊樣板來判斷一個程式是否需要受監督。當一個需要受監督的程式開始執行，Wrapper Manager會動態地產生監督此應用程式的FSM實體，開始攔截此程式發出的所有系統呼叫、執行前置與後置動作(Pre-actions&Post-actions)、檢查是否需要進行狀態轉換。若此應用程式會執行fork系統呼叫，則由合作攻擊處理模組產生出家族FSM實體。若此程式欲執行會導致FSM實體或家族FSM實體轉換到最終狀態的系統呼叫，則立即停止此應用程式的執行，阻止其對系統造成傷害。系統之運作流程如圖28所示。

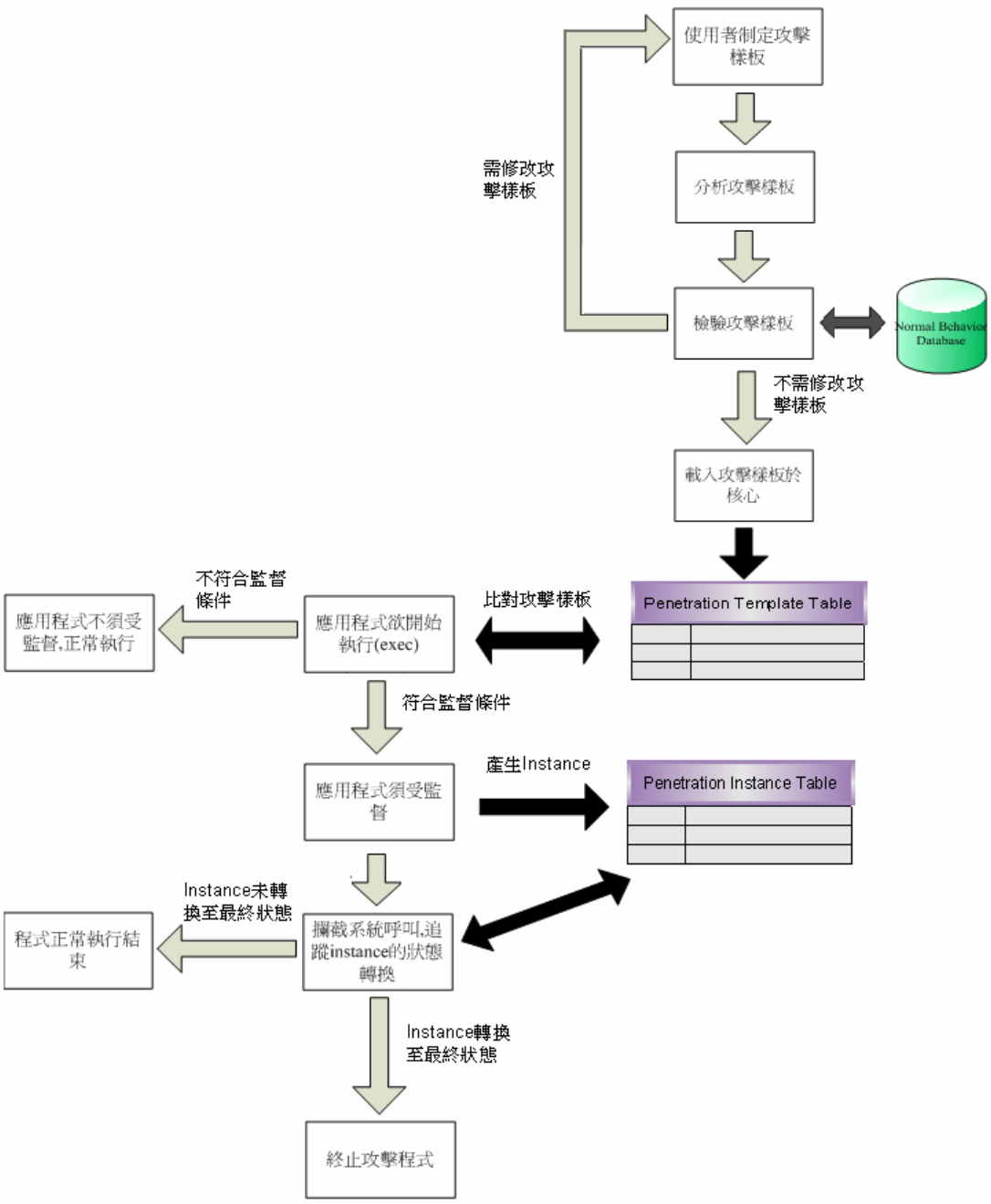


圖 28 系統運作流程

5.2 使用者層模組(User level module)

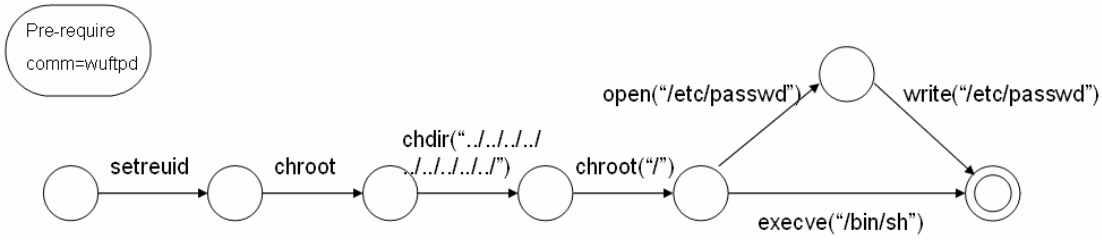
使用者層模組的主要功能在於對攻擊樣板的前置處理，包含提供介面給使用者制定攻擊樣板、分析使用者定義之攻擊樣板，以抵抗插入 no-op 系統呼叫之攻擊、檢驗使用者定義之攻擊樣板，防止使用者使用不當攻擊樣板監督程式，造成高誤判率。這些模組的功能將於本節中介紹。

5.2.1 樣板制定介面(State-based Rule configuration interface)

在本系統中，我們可把複雜的入侵規則用簡單直覺的圖形(有限狀態機)來表示。樣板制定介面提供以滑鼠拖拉方式的GUI圖形介面給使用者來制定攻擊樣板。例如圖29是一個利用wuftpd FTP server的漏洞取得root權限並進行攻擊之樣板。監督的條件可以檢查程式執行時的三個欄位uid，gid，comm來決定，也可以指定all，即所有程式都要監督。

這個例子指定只要是comm=wuftpd的程式都必須監督，攻擊樣式如圖所示，若依序執行了這幾個的系統呼叫： setreuid，chroot，chdir(“../../../../../../../../../../”)，chroot(“/”)，便可取得root權限，並逃脫chroot的限制，接著進行攻擊。使用者在樣板制定介面制定的攻擊樣板，會交由樣板分析與檢驗模組進行分析與檢驗。

keyword	Condition	expression
All		
user	[= | !=]	***User-name***
group	[= | !=]	***Group-name***
comm	[= | !=]	***Executed-program-name***



`圖 29 攻擊樣板

5.2.2 樣板分析與檢驗模組(Template analysis and testing module)

爲了在使用者定出不良的攻擊樣板時提出警告，進而減少誤判，且爲了阻止有心的入侵者刻意插入no-op系統呼叫來規避偵測，我們必須對使用者自行定義的攻擊樣板(user define template)，稱原始攻擊樣板，進行分析和檢驗。此部分之

功能由樣板分析模組(Template analysis module)與樣板檢驗模組(Template testing module)來實現。

樣板分析模組先對原始攻擊樣板以圖22的演算法進行分析，視需要將其拆爲若干個sub-FSM，再利用圖24之演算法將每個sub-FSM轉爲 ε -NFA，最後轉成真正用於偵測攻擊的攻擊樣板(final Tempalrte)。例如在圖29中使用者所制定的原始攻擊樣板，在經過樣板分析模組的分析與轉換後，會轉爲如圖30的最終攻擊樣板。

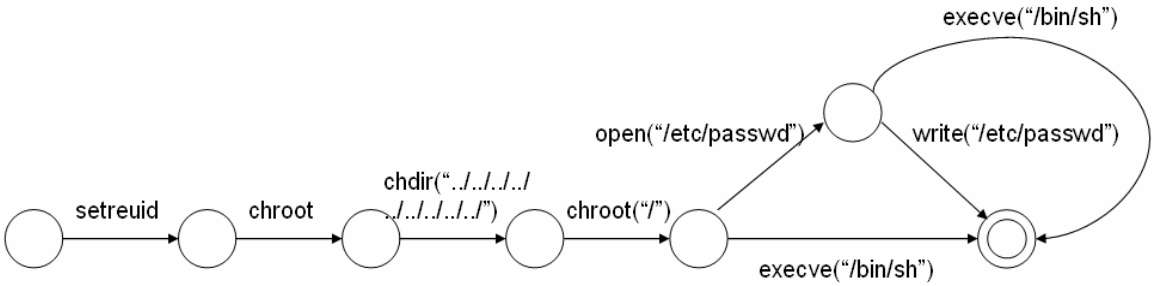


圖 30 經轉換之最終攻擊樣板

攻擊樣板經過分析後，還必須經過檢驗才能載入核心中。樣板的檢驗由樣板檢驗模組從正常行爲資料庫中，取出代表正常行爲的系統呼叫順序，測試是否會讓此攻擊樣板轉換到最終狀態，如圖31。

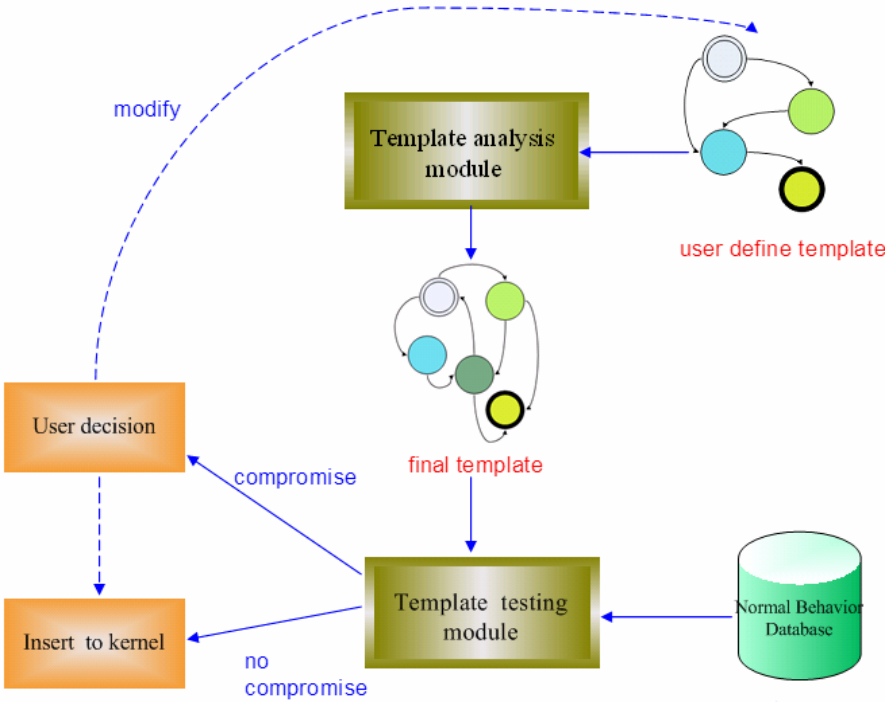


圖 31 樣板分析與檢驗模組

若所有代表正常行爲的系統呼叫順序皆不會使此攻擊樣板轉換到最終狀態，表示此攻擊樣板不會把這些正常行爲誤判成攻擊，因此可以直接載入核心中。若有代表正常行爲的系統呼叫順序會使此攻擊樣板轉換到最終狀態，表示此攻擊樣板可能是有問題的，可能會造成誤判，此時必需提報使用者，決定是否要修改攻擊樣板。

樣板通過檢驗後，接著便由Wrapper Driver載入核心，由核心層模組根據這些使用者制定之樣板來監督程式的執行，偵測是否有符合的攻擊行爲。

5.3 核心層模組(Kernel level module)

核心層模組的主要功能在於即時地偵測攻擊，包含產生 FSM 實體、攔截系統呼叫、執行攻擊 FSM實體的狀態轉換，處理合作攻擊。另外訓練正常行爲資料庫也是核心層模組的功能之一，這些模組的功能都將於本節中一一介紹。

5.3.1 正常行為收集模組(Normal Behavior Collector)

樣板檢驗模組中使用的正常行爲資料庫，是由使用者在安全的環境下，由正常行爲收集模組記錄正常使用下之系統呼叫順序所建立出來的。使用此方式建立正常資料庫的前提是系統必須處於安全、無攻擊之虞的環境下，蒐集到的正常資料庫才有意義，檢驗攻擊樣板時才能精確地檢查出錯誤。

此訓練方式適用於任何應用程式，使用者可透過Wrapper Driver，在Wrapper Manager中設定紀錄正常系統呼叫的條件，例如指定收集某個程式執行的系統呼叫。Wrapper Manager即可根據設定的條件呼叫正常行爲收集模組爲該應用程式訓練出專屬的正常行爲資料庫，如圖32所示。訓練資料庫的時間和資料庫的大小亦可由使用者自行決定，訓練時間越久、資料庫越大，越能精確地檢驗攻擊樣板、減少誤判情形。另外在測試階段時，若使用者發現正常行爲資料庫中有不合實用的資料，亦可對此資料庫進行更新的動作。

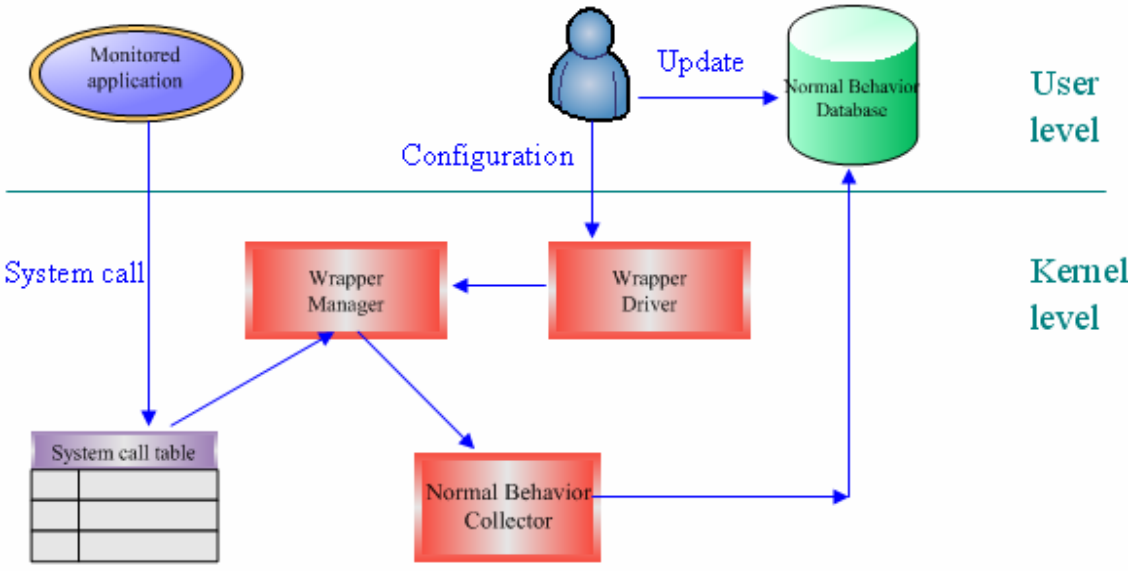


圖 32 正常行爲資料庫之建立

5.3.2 合作攻擊處理模組(Co-operation handle module)

當我們的Wrapper Manager攔截到一個exec系統呼叫，經過與攻擊樣板的監督條件比對，可以判斷是否要監督發出此exec的程式。若程式需要受監督，Wrapper Manager會產生出監督這個程式的FSM實體。當這個程式執行fork系統呼叫，合作攻擊處理模組會產生出子程式的FSM實體，同時也會產生出一個家族FSM實體，如圖33所示。之後攔截到父程式執行的系統呼叫，合作攻擊處理模組都會檢查父程式本身和家族的FSM實體。攔截到子程式執行的系統呼叫，則會檢查子程式本身和家族的FSM實體，如圖34所示。

若是子程式再執行fork系統呼叫，則合作攻擊處理模組會產生出孫程式的FSM實體，但不會再產生新的家族 FSM實體，而是會將孫程式的FSM實體和已產生的家族FSM實體建立起關連，屬於同一個監督家族，如圖34。

由於家族成員都是由最早的一個程式所fork出來的，彼此間有合作進行攻擊的可能。因此在家族FSM產生後，家族成員中的任何一個程式執行系統呼叫都會參考家族FSM的轉換條件，故家族成員間彼此合作進行攻擊時，不管哪個家族成員執行攻擊的步驟，都會使得家族FSM做狀態轉換，而無法逃過我們的偵測。家族FSM實體必須等到所有家族成員皆執行結束，才會被系統刪除。

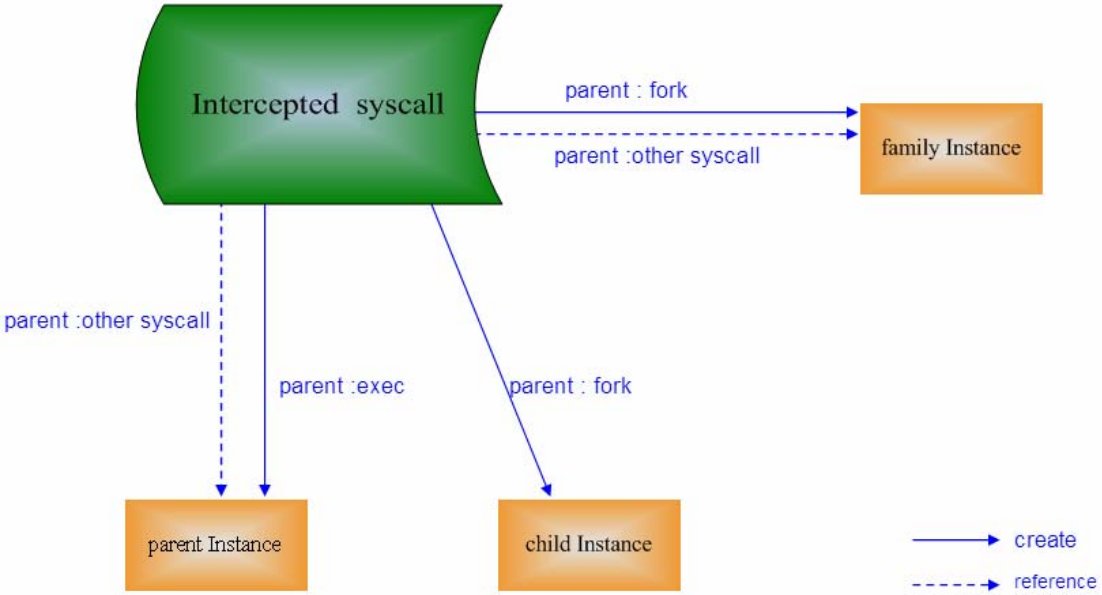


圖 33 合作攻擊處理模組

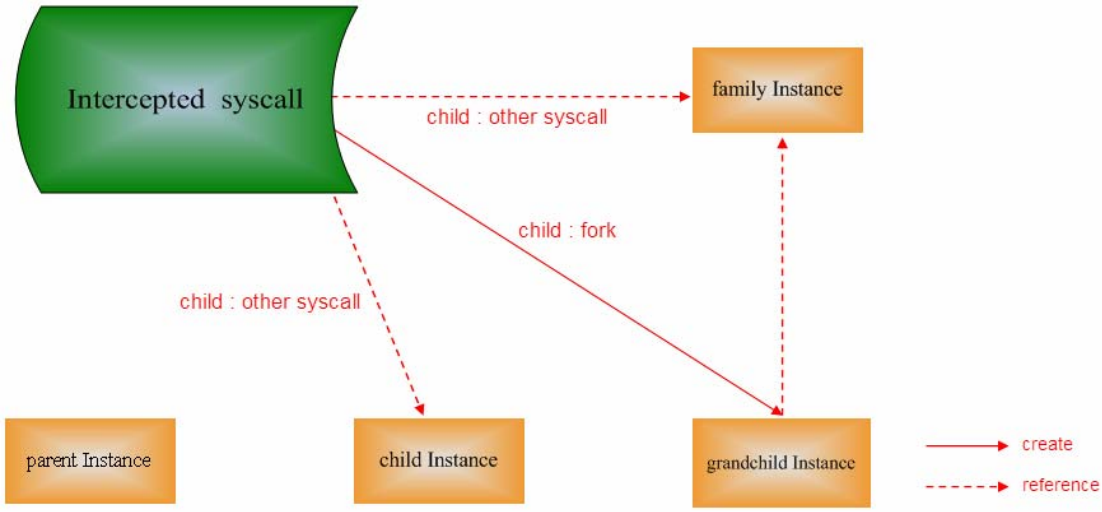


圖 34 合作攻擊處理模組

5.3.3 其他核心層模組

本節中我們將介紹其他核心層模組之功能以及各模組間的關係，包含Wrapper Manager，Wrapper Driver，State Information center，這些模組皆以Wrapper Manager爲中心來運作。

Wrapper Manager

當一個應用程式開始執行時，Wrapper Manager會檢查所有攻擊樣板中的監督條件是否與此應用程式相符，如程式的uid、gid、command等等。例如圖29之

攻擊樣板，經由Wrapper Driver載入後便存在核心的攻擊樣板表(penetration template table)中，其監督條件爲comm=wuftpd，因此任何人只要執行wuftpd這個程式，便會在攻擊樣板表中找到符合監督條件comm=wuftpd之攻擊樣板圖29。此時Wrapper Manager會動態地產生監督wuftpd的FSM實體，存在攻擊FSM實體表(penetration instance table)中。當wuftpd執行結束，即發出exit系統呼叫時，Wrapper Manager便會將對應的FSM實體從攻擊FSM實體表中刪除。

Wrapper Manager根據FSM實體的目前狀態與轉換條件，檢查每個所攔截之系統呼叫是否會使對應的FSM實體轉換到最終狀態。Wrapper Manager攔截到系統呼叫，在呼叫原本核心中的系統呼叫函式之前，可先執行某些動作，例如設定某些變數供觀察，此爲Pre-actions。若執行系統呼叫後不會使FSM實體轉換到最終狀態，則允許執行此系統呼叫。真正執行系統呼叫後，可再做一次執行結果的檢查(Post-actions)，看是否需要將FSM實體轉換到下一個狀態。若是執行此系統呼叫會導致轉換到最終狀態，表示偵測到攻擊行爲，則立即停止此應用程式的執行。

Wrapper Manager的另一個功能是根據使用者設定的條件，呼叫正常行爲收集模組，記錄下正常使用狀況下的系統呼叫順序，存到所訓練的正常行爲資料庫中。

Wrapper Driver

Wrapper Driver是使用者層元件與核心層元件溝通的橋樑，提供各種命令給使用者層元件呼叫。使用者可利用這些命令設定訓練正常行爲資料庫的條件，亦可管理、載入攻擊樣板於核心。

State Information center

State Information center是核心元件中儲存資訊的地方，包含儲存被載入之攻擊樣板的攻擊樣板表，儲存FSM實體之攻擊FSM實體表，目前所監督之程式列表，目前所監督之系統呼叫列表，以及各個FSM實體的目前狀態。

第六章、實驗與評估

在這個章節，我們將做一些實驗來證明我們提出之系統的效能與可行性。6.1介紹實驗的環境，包含使用的硬體、作業系統以及實驗的方法。有效率的入侵偵測防禦系統在即時偵測攻擊時，必須避免對系統造成太大的負擔，6.2中將討論我們的系統執行時間上的負擔(overhead)。6.3中介紹實際利用使用者訓練出來的正常行為資料庫來檢驗出不當攻擊樣板的實例。6.4中，我們使用蓄意規避偵測的攻擊行為，證明我們的系統可以防止這些規避行為，更為精確地偵測出攻擊。

6.1 實驗環境與實驗方法

在介紹各個實驗結果之前，我們先介紹底下各節所使用到的實驗環境。實驗所使用的硬體方面，CPU為Intel Pentium-III 1.1 GHz的CPU，有512MB的SDRAM。在軟體方面，使用的作業系統是Debian Linux，核心版本為2.4.28，C語言編譯器為gcc、g++。

本章的所有實驗，皆是將我們的系統安裝到上述環境中來執行。在6.2中的各項測試執行時間負擔(runtime overhead)之實驗，皆是在安裝系統與不安裝系統的情形下，各執行測試程式500次所測得的執行時間取平均值。6.3的攻擊偵測實驗，則是以C語言撰寫會執行特定之攻擊系統呼叫順序(penetration system call sequence)的程式，實驗是否會被我們的系統偵測出來。

6.2 執行時間負擔(Runtime overhead)

本節將評估我們AMA-IPS系統之時間負擔(runtime overhead)。本系統的主要時間負擔(runtime overhead)來自於狀態的轉換。我們設計第一個測試程式為一個複製檔案的程式，開啓一個文字檔並將內容複製到另一個檔案，程式執行時每個read，write系統呼叫皆會造成監督此程式之FSM實體的狀態轉換，得到結果如圖

35。不論複製的檔案大小多大，系統花在狀態轉換的時間爲固定值，約爲1300 μs，因此當複製的檔案越大，程式花在作I/O的時間越長，則花在做狀態轉換上的比例越小，整體的時間負擔(runtime overhead)便會越小，如表4所示。

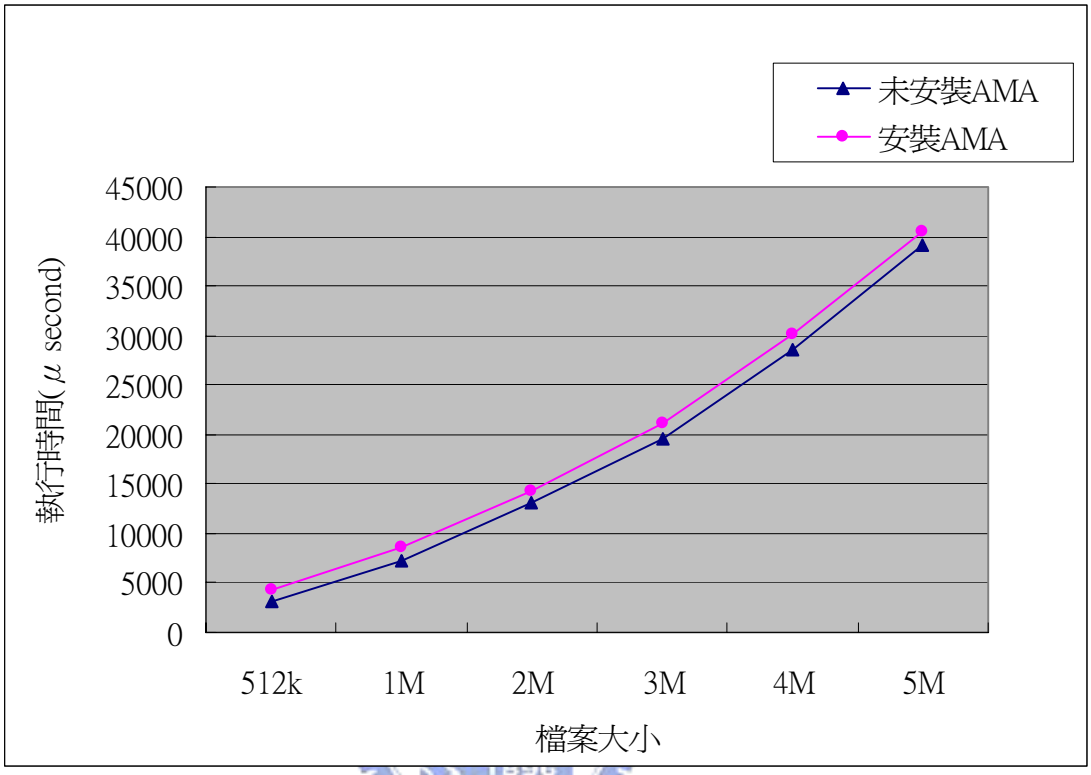


圖 35 檔案大小與執行時間關係圖

表 4 時間負擔(runtime overhead)關係表

檔案大小	512k	1M	2M	3M	4M	5M
增加的時間負擔(runtime overhead)百分比	35.8%	17.7%	10.6%	8.3%	5.5%	3.5%

然而第一個測試程式與攻擊樣板的組合可說是最差狀況。通常使用者在制定攻擊樣板時，攻擊樣板上的關鍵系統呼叫都是極少發生的，因爲一般程式執行時，跟正常行爲比起來，攻擊行爲的比率非常低，並不會像上例一樣頻繁的做狀

態轉換。因此在第二個測試程式中，我們將實驗複製2M檔案之程式，改變其行為條件與攻擊樣板，降低系統呼叫中會造成樣板狀態轉換之系統呼叫所佔的百分比，結果如圖36。可以看出造成樣板的狀態轉換之系統呼叫所佔的百分比在10%以下時，時間負擔(runtime overhead)幾近於0。

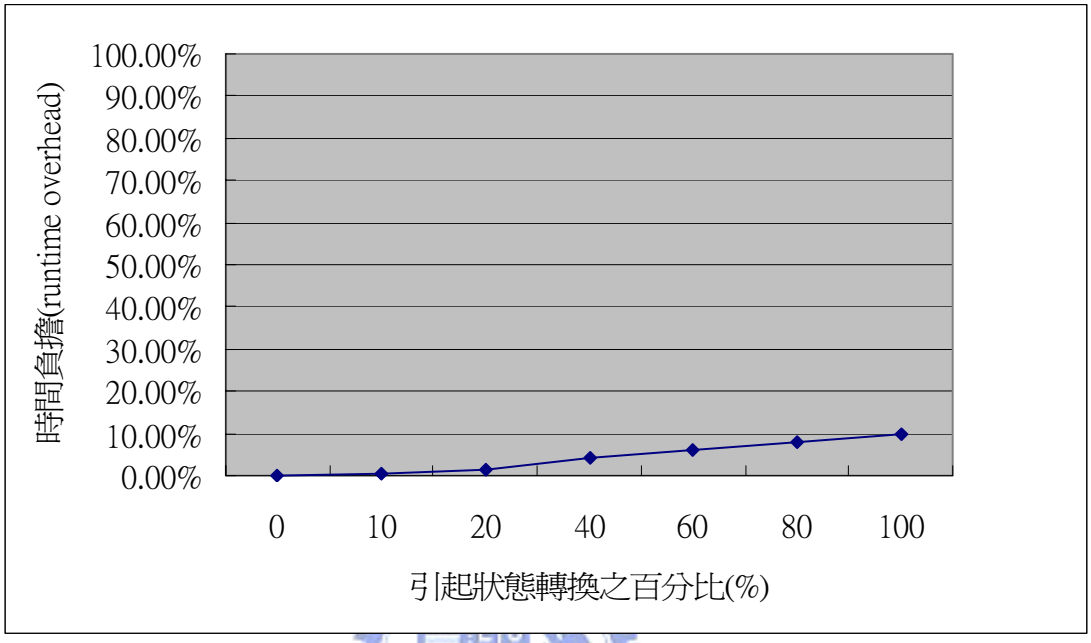


圖 36 狀態轉換百分比與時間負擔(runtime overhead)關係圖

一般程式正常執行時，狀態轉換的百分比是很低的，我們以實際上的應用程式作實驗，此程式為一個搜尋檔案內資訊、並做分析統計的perl程式，因為大部分的時間在做CPU的運算，屬於CPU-Bound類型的程式，甚少執行系統呼叫、意即甚少發生FSM實體的狀態轉換。其結果在無安裝本系統的情況下，平均執行時間為4.982681秒，而載入本系統後的平均執行時間為4.954225秒，時間負擔(runtime overhead)只有0.5%。

以上實驗皆是系統中只有單一程式被單一攻擊樣板監督的情形，接著我們以複製2M檔案之程式，實驗攻擊樣板數目增加與同時執行多個受監督之程式的情形。圖37為單一程式被多個攻擊樣板監督之關係圖，可看出即使程式符合多個不同的攻擊樣板之監督條件，同時被多個攻擊樣板監督，只要程式仍是正常執行，對於執行時之時間負擔(runtime overhead)仍能維持在一定的平穩狀態，不會因同時符合多個監督條件而造成程式執行的效率變差。這是因為即使程式符合多個監督條件而受監督，但在正常執行下並不會造成多個樣板同時做頻繁的狀態轉換，因此可維持穩定的低執行時間負擔(runtime overhead)。

圖38爲同時執行數個需被監督的程式時，時間負擔(runtime overhead)之變化情形。可看出同時執行多個受監督的程式時，執行時間負擔(runtime overhead)亦能維持穩定，主要原因是攔截到某個程式發出的系統呼叫時，尋找監督此程式的FSM實體是採用hash的方式來搜尋，因此同時多個程式執行只會增加尋找對應之FSM實體的時間，這部份的時間非常少，因此可維持穩定的低執行時間負擔(runtime overhead)。

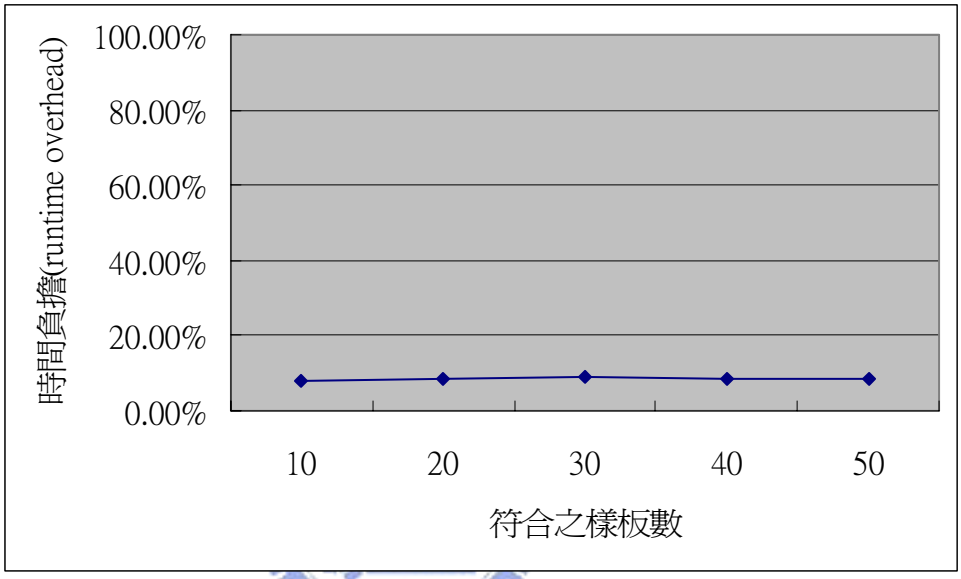


圖 37 監督樣板數與時間負擔(runtime overhead)關係圖

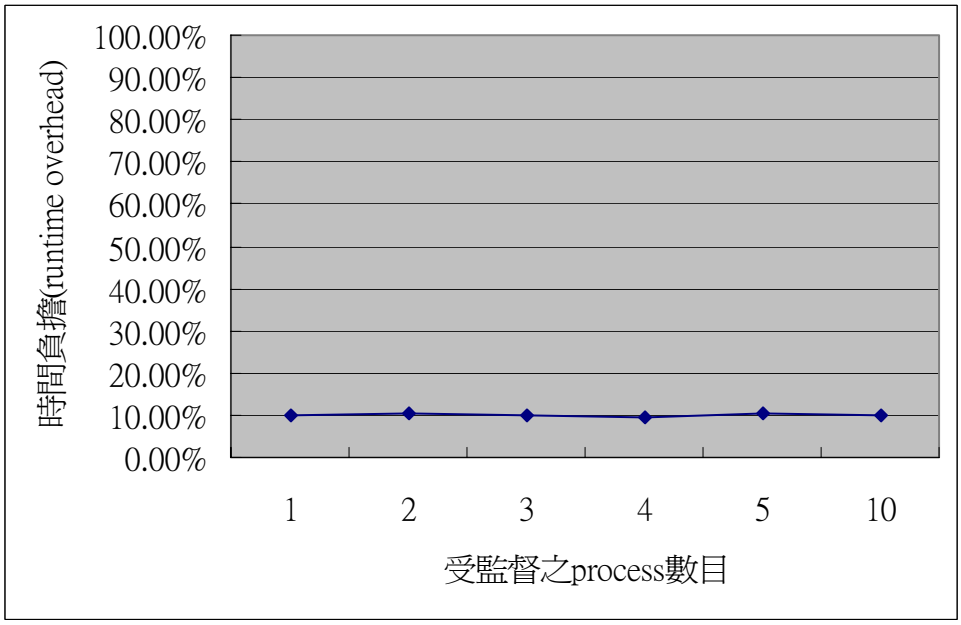


圖 38 process 數目與時間負擔(runtime overhead)關係圖

最後我們實驗同時多個程式執行，且每個程式皆被多個攻擊樣板所監督的情形，如圖39。圖中每一線段代表同時執行之程式數目，可見在如此複雜的監督情形下，我們的系統仍能維持穩定的低時間負擔(runtime overhead)，並不會因程式數目增多與樣板數增加而造成執行效率不彰。

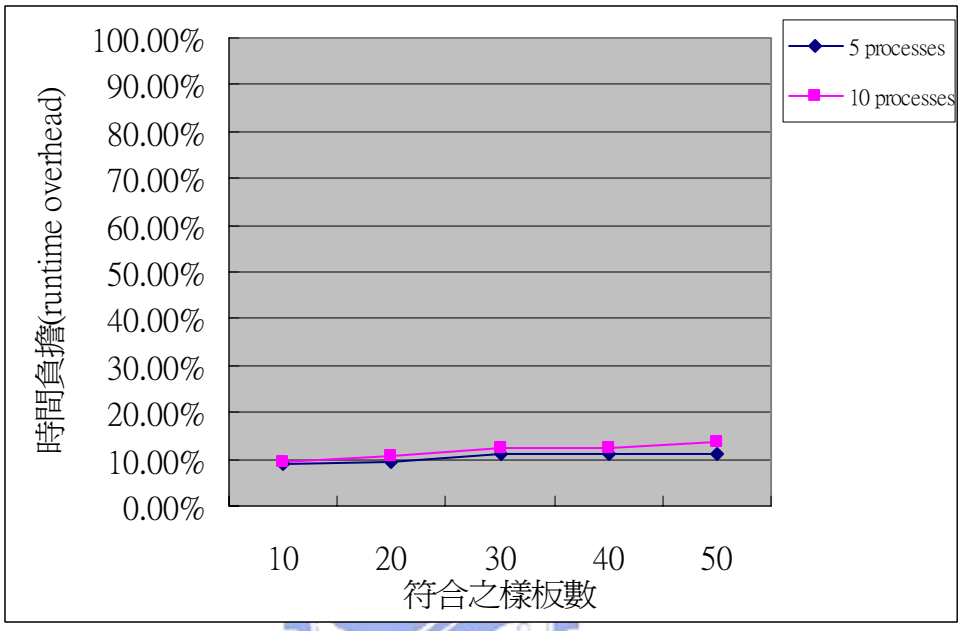


圖 39 受監督程式數與樣板數之關係圖

6.3 攻擊偵測實驗

本節中，我們將實驗證明本系統對兩種規避IDS偵測的攻擊方式之偵測能力，即防止插入no-op系統呼叫之攻擊與程式間的合作攻擊。

6.3.1 插入 no-op 攻擊實驗

首先，我們假設一個攻擊的系統呼叫順序如下：

setreuid(0,0) , open(“/etc/passwd”) , write(“/etc/passwd”)

在pH IDS的偵測方式中，假設window size爲3，我們只要在其中插入一個no-op的系統呼叫，例如read，使得攻擊的系統呼叫順序如下：

setreuid(0,0), open(“/etc/passwd”) , read(“/etc/passwd”),write(“/etc/passwd”)

便可以逃過pH IDS的偵測，因爲當window size爲n時，插入若干的no-op的系統呼

叫使整個攻擊的系統呼叫順序長度大於n，便可以規避pH IDS的偵測。

我們的系統基於STAT之特性，假設攻擊者已知使用者會定義的攻擊樣板，如圖40，但我們的FSM一定要遇到關鍵系統呼叫才會做狀態的轉換，因此插入不相干的no-op系統呼叫並不會影響我們的偵測。如上例在做過setreuid、open這兩個關鍵系統呼叫後，FSM停留在狀態3，此時插入read這個無關的系統呼叫對我們的偵測並無影響，我們的FSM會一直停留在狀態3直到攔截到write這個系統呼叫才會轉換到最終狀態，偵測出攻擊。

此外，有經驗的攻擊者可能會利用STAT的弱點，試圖插入攻擊樣板上的其他系統呼叫來進行繞路攻擊，其攻擊系統呼叫如下：

setreuid(0,0) , open(“/etc/passwd”) , chroot(“\”)

其中底線部分是關鍵系統呼叫，open爲no-op系統呼叫，主要試圖利用open來做繞路的攻擊。先執行setreuid、open使得圖38的FSM走到狀態3，接著利用FSM在狀態3不會攔截chroot的缺點，完成setreuid、chroot這個攻擊系統呼叫順序而不被偵測出來。但在我們的系統中，我們的樣板分析模組會將圖40的原始攻擊樣板，經過4.2中的各項步驟，轉換成抵抗no-op攻擊之攻擊樣板，如圖41所示。

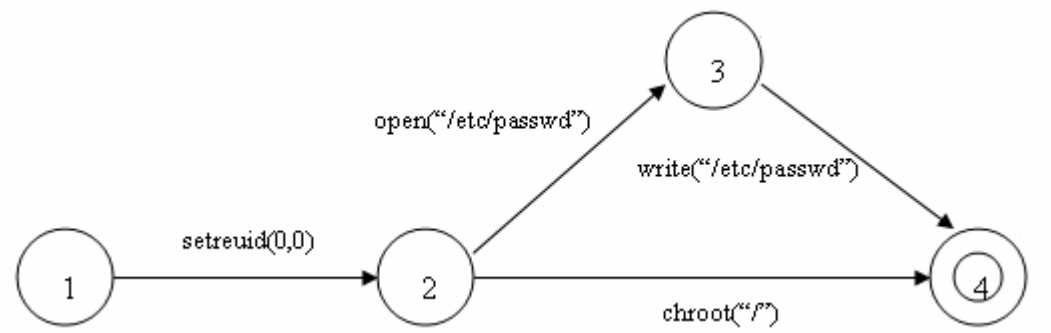


圖 40 原始攻擊樣板

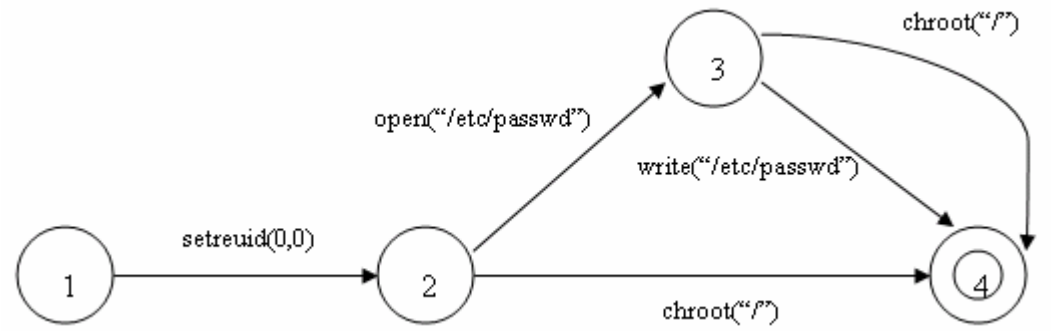


圖 41 可抵抗 no-op 攻擊之攻擊樣板

在圖41最終攻擊樣板中，我們可以偵測出上述插入no-op系統呼叫的繞路攻

擊，如圖42，且無論攻擊者對原始攻擊樣板插入多少no-op的系統呼叫，或嘗試做繞路攻擊，我們都可以成功地偵測出來。

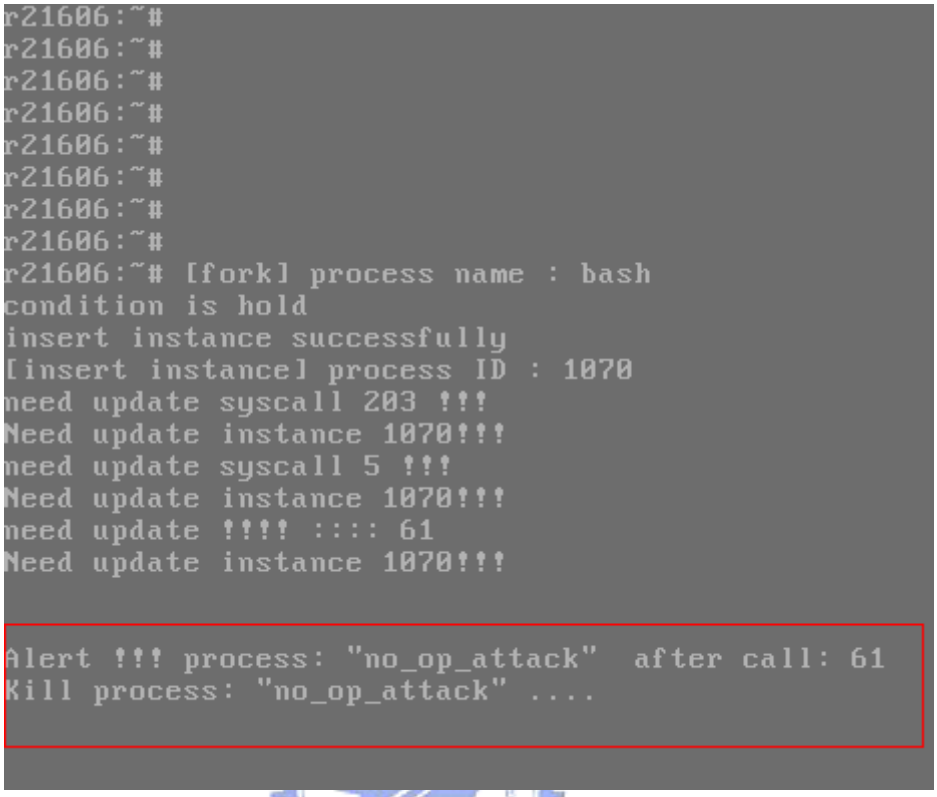


圖 42 偵測 no-op 攻擊

6.3.2 程式間合作攻擊實驗

在其他入侵偵測系統，如傳統STAT的偵測中，是以單一程式爲單位進行監督，並無處理程式間合作攻擊的機制，因此攻擊者有機會利用程式間彼此合作來進行攻擊入侵，以規避入侵偵測系統的偵測。在這個實驗裡，我們假設攻擊者已知使用者定義的攻擊樣板，如圖43。攻擊者可將攻擊的系統呼叫拆開成幾個部份，分別讓不同的程式執行，例如將攻擊系統呼叫拆開如下：

fork前：setreuid(0,0)

fork後：

parent process：open(“/etc/passwd”)

child process：setreuid(0,0) ， write(“/etc/passwd”)

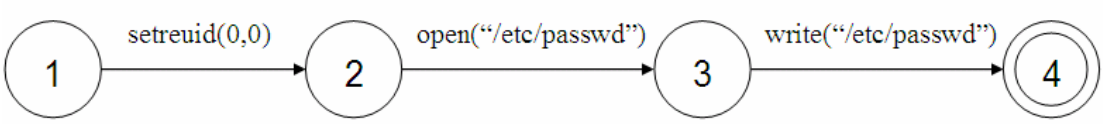


圖 43 合作攻擊實驗之樣板

若以STAT監督父程式的執行，只會觀察到其作了setreuid後轉換到狀態2，因爲父程式並不會執行write，因此不會轉換到狀態3而被判斷爲攻擊。在子程式的部份，以STAT來監督，可觀察到子程式在setreuid後便停留在狀態2，接受由父程式所做的open得來的檔案描述子，而不需要自己做open的動作，因此停留在狀態2的子程式即使執行了write也不會轉換到最終狀態而被判斷爲攻擊。所以在傳統STAT的監督下，不管是父程式或是子程式，都不會讓圖43的攻擊樣板走到最終狀態，因此STAT無法偵測到此合作攻擊。

在我們的系統中，有合作攻擊處理模組在父程式執行fork時便建立起家族FSM實體，此家族FSM實體同時監督父程式與子程式。本實驗中，由父程式執行setreuid後家族FSM實體轉換到狀態2，再由父程式的open而轉換到狀態3，最後攔截到子程式的write，便會轉換到最終狀態而偵測到攻擊，如圖44所示。

```
r21606:~#
r21606:~#
r21606:~#
r21606:~# [fork] process name : bash
condition is hold
insert instance successfully
[insert instance] process ID : 1155
need update syscall 203 !!!
Need update instance 1155!!!
need update syscall 5 !!!
Need update instance 1155!!!
[fork] process name : co_op_attack
insert instance successfully
[insert instance] process ID : 1156
insert instance successfully
[insert instance] process ID : 100000
need update syscall 4 !!!
Need update family instance 100000!!!


Alert !!! process: "co_op_attack"  after call: 4
Kill process: "co_op_attack" ....
```

圖 44 偵測合作攻擊

6.4 不當樣板之預警

Misuse入侵偵測技術造成誤判的原因主要來自於使用者定義的攻擊樣式模糊不清，其中可能包含到正常行爲。本節中將實驗利用在正常使用狀況下紀錄的正常行爲，幫助使用者檢驗出可能有問題之攻擊樣板，藉以降低偵測時產生誤判的機率。

假設我們欲對sftp-server這個程式進行監督，爲了防止日後定出錯誤、易產生誤判的攻擊樣板，在訓練階段，我們利用Normal Behavior Collector經過爲期兩天的正常行爲訓練，產生出約2M大小之正常行爲資料庫。此訓練時間我們只使用內部網路(Intranet)進行連線登入操作，以確保不會發生攻擊行爲。

接著我們試圖定義一個攻擊樣板如圖45，目的在防止攻擊者利用sftp-server可能存在的漏洞取得root權限後，留下後門程式對系統造成傷害。

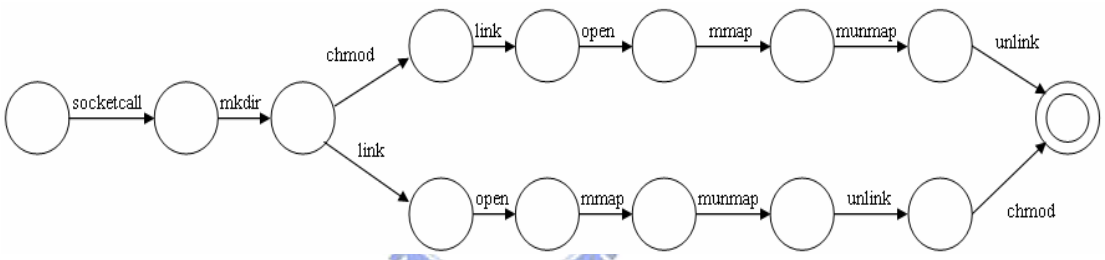


圖 45 sftp-server 不當攻擊樣板

我們定義的攻擊樣板是爲了防止利用漏洞取得root權限的攻擊者任意產生新目錄、更改權限、寫入可能的入侵程式。然而這樣定義出來的攻擊樣板經過樣板檢驗模組使用訓練出來的正常行爲資料庫進行檢驗後，發現在訓練階段正常使用的狀況下也會使得此樣板走到最終狀態，如圖46。

因爲圖45的系統呼叫順序，在一個合法的使用者，於其擁有相對權限之目錄下操作時，也有可能產生。因此圖45的攻擊樣板不夠嚴謹，會將正常的行爲判斷成攻擊，算是不良之攻擊樣板，用來偵測攻擊會產生誤判。

我們的系統可在使用者制定好樣板時馬上做檢驗，提出警報給使用者，以避免使用者將不當攻擊樣板拿來做偵測而導致誤判過多的情形。我們也藉此進而有效降低誤判率。

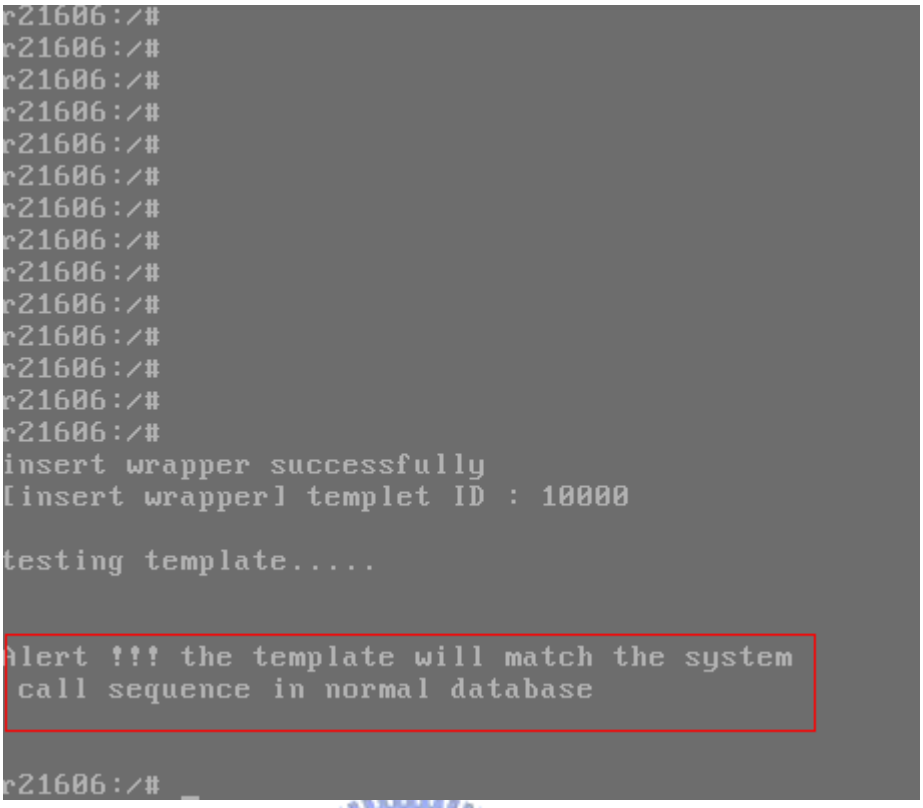


圖 46 不當攻擊樣板之檢驗

第七章、討論與結論

在第三章相關研究中，我們介紹過幾種入侵偵測防禦系統，它們各有其優缺點，在這個章節，我們將討論在IDS系統的各項特性中，本論文提出的系統與其他相關研究之比較，並對本論文做一個總結。7.2中，我們提出幾點未來可以繼續擴充系統功能的研究方向。

7.1 討論與結論

首先針對幾項IDS系統的特性，將本論文提出的系統與其它IDS系統做一個比較，如表5。

表 5 與其他 IDS 系統之比較表

	Our system	pH IDS	IGSTAM	STAT
Real-time intrusion prevention	YES	YES	NO	NO
Graphical rule configuration	YES	N/A	N/A	YES
Transparency	YES	YES	YES	YES
Resist no-op attack	YES	NO	NO	NO
Resist co-operation attack	YES	NO	NO	NO
Resist equivalent attack	NO	YES	YES	NO

Real-time intrusion prevention表示一個IDS系統能否在攻擊對系統造成傷害前阻止其發生，IGSTAM屬於入侵偵測系統，只能做事後的入侵偵測。我們的系

統攔截所有系統呼叫，可在FSM實體走到最終狀態前停止攻擊程式，防止對系統造成傷害。

Graphical rule configuration 表示系統是否提供方便的圖形化的攻擊定義方式，我們的系統與STAT都提供圖形化介面，以畫FSM的方式來描述攻擊，pH與IGSTAM是anomaly的IDS系統，並不需要自行定義攻擊。

Transparency表示受監控的應用程式是否會察覺自己已經遭受監控，或者在程式行爲上是否需要做改變來配合監督程式。我們的系統在核心層攔截系統呼叫，因此使用者層的應用程式不會察覺自己受到監控。其他系統也都是在核心層進行監控。

Resist no-op attack則代表能否抵抗插入no-op的攻擊，我們的系統會對使用者定義的原始攻擊樣板進行分析處理，可防止插入no-op的攻擊。IGSTAM與pH IDS都是以pattern matching的方式偵測攻擊，無法防止插入no-op的攻擊，而STAT則是會有繞路攻擊的問題。

Resist cooperaton attack表示能否抵抗程式間的合作攻擊，除了我們的系統使用監督家族的觀念來防止合作攻擊之外，其他的系統並沒有提出關於此種攻擊的解決方法。

Resist equivalent attack則代表能否抵抗已知攻擊的變種，即攻擊者修改攻擊樣式來達成相同的攻擊效果。我們的系統目前只能防禦已知的攻擊樣式，若攻擊樣式不在定義的範圍則無法偵測出來，這部分在未來工作可以進行加強。pH和IGSTAM屬於anomaly的IDS系統，若攻擊的變種與正常行爲差異大，還是有可能被偵測出來。

總結我們提出的AMA-IPS系統，有下列幾項優點：

(1) 有核心層wrapper的優點。在核心層進行監控，執行時間的負擔較低，任何應用程式要向系統要求資源都一定會被我們所監督，且可以在造成傷害之前阻擋攻擊，達到及時防護功能。

(2) 有STAT狀態轉換描述的優點。我們的系統提供圖形化描述攻擊的介面，使用狀態轉換描述的好處是使用上比較直覺簡單，可以用直覺的方式描述出複雜的入侵行爲。

(3) 減少警報誤判(false positive)。我們用基因演算法中的負向選擇概念，利用訓練出來的正常行爲資料庫檢驗攻擊樣板，防止使用者定出錯誤、易

產生誤判的攻擊樣板，降低發生誤判的機會。

(4) 克服規避IDS偵測的技巧。我們的系統透過對事先對原始攻擊樣板的分析以及動態建立監督家族，使得插入no-op的攻擊以及程式間的合作攻擊無法實現。

7.2 未來工作

本論文提出一個即時的入侵偵測防禦系統，克服了某些現行入侵偵測系統的缺點，但仍有些部份可以做更進一步的研究與改進，以將整個系統擴充得更完整，底下我們列出一些可改進的地方。

(1) 克服其他規避IDS偵測的技巧。我們希望能夠再克服其他規避IDS偵測的技巧，包括等價攻擊與緩衝區溢位(Buffer overflow)。

所謂的等價攻擊，即攻擊者修改已被知道的攻擊樣式，使用具有相同作用的系統呼叫來達成攻擊的目的，例如使用mmap()取代read()。我們希望能夠找出這些具有相同作用的系統呼叫，以便使用者定義其中一種攻擊樣板時，也能有效防止等價的攻擊。

緩衝區溢位是許多入侵行為的起點，但在進行緩衝區溢位時並不會調用到特別的系統呼叫，因此只監督系統呼叫無法偵測出來，必須要用其他的方法來偵測。

(2) 結合資料探勘(data mining)的技術，自動地產生攻擊樣板。我們的系統使用系統呼叫順序(system call sequence)來描述攻擊行為，然而使用者即使了解某應用程式的漏洞，或者怎樣操作屬於非法動作，但卻可能因對系統呼叫的認識有限，無法直接以系統呼叫順序來描述出攻擊。

我們希望使正常行為收集模組(Normal Behavior Collector)兼具收集異常行為的功能，讓使用者可直接操作不合法行為，由此模組記錄下此異常行為之系統呼叫順序。接著可利用資料探勘的技術，將異常行為與正常行為經過比對分析，找出代表異常行為之規則，也就是代表異常行為之關鍵系統呼叫順序，並自動將其轉換為可偵測此異常行為之攻擊樣板。

參考文獻

[1] A. Somayaji, S. Forrest, "Automated Response Using System-Call Delays," in *Proceding of 9th Usenix Security Symposium*, 2000, pp.185.

[2] Bai, Y., Kobayashi, H., "Intrusion Detection Systems: technology and development," in *Proceding of 17th International Conference on 27-29 March*, 2003 Page(s):710 – 715.

[3] Caberera, J.B.D., Ravichandran, B., Mehra, R.K., and Sci. Syst. Co., Woburn, "Statistical traffic modeling for network intrusion detection," in *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2000, pp 466-473.

[4] D. Wagner and P. Soto ., " Mimicry Attacks on Host-Based Intrusion Detection Systems," in *Proceding of ACM Conference on Computer and Communications Security*,2002, pages 255-264

[5] Dozier, G.,Brown, D., Hurley, J., Cain, K, "Vulnerability analysis of AIS-based intrusion detection systems via genetic and particle swarm red teams," Evolutionary Computation, 2004. CEC2004. Congress on Volume 1, 19-23 June 2004 Page(s):111 - 116 Vol.1.

[6] Eskin, E., Wenke Lee, Stolfo, S.J., "Modeling system calls for intrusion detection with dynamic window sizes," in *Proceding of DARPA Information Survivability Conference & Exposition II*, 2001. DISCEX '01. Proceedings Volume 1, 12-14 June 2001 Page(s):165 - 175 vol.1.

[7] F. Besson, T. Jensen, D. L. Metayer, and T. Thorn., "Model checking security properties of control flow graphs," *Journal of Computer Security*, 9:217-250, 2001.

[8] F Gonzalez and D Dasgupta, "Anomaly detection using real-valued negative selection," *Journal of Genetic Programming and Evolvable Machines*, 4:383--403, 2003.

[9] Feng, H.H., Kolesnikov, O.M., Fogla, P., Lee,W., Gong, W., "Anomaly Detection Using Call Stack Information," in *Proceedings of the 2003 IEEE*

Symposium on Security and Privacy,Berkeley, p62, *CA* (2003)

[10] Ghosh, A.K., Wanken, J., Charron, F., "Detecting anomalous and unknown intrusions against programs," in *Proceedings of the 14th Annual Computer Security Applications Conference*, 1998.,pp. 259-267.

[11] Iguchi, M., Goto, S., "Network surveillance for detecting intrusions," Internet Workshop, 1999. IWS 99 ,pp 99-106.

[12] Joseph, M. McAlerne and Stuart Staniford, James A. Hoagland, "Practical Automated Detection of Stealthy Portscans," Silicon Defense Publications, http://downloads.securityfocus.com/library/spice-ccs2000.pdf

[13] K.M.C. Tan, K.S. Killourhy, R.A. Maxion, "Undermining an Anomaly-Based Intrusion Detection System Using Common Exploits," to appear at *RAID* 2002 Page(s) : 54-73, 16-18 Oct. 2002

[14] Koral Ilgun, Richard A. Kemmerer, and Phillip A. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection Approach," *IEEE Transaction on Software Engineering*, Page(s): 181-199, Vol.21, No.3, March 1995.

[15] Phillip A. Porras, "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)*," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, California*, MAY 9-12, 1999.Page(s): 146-161

[16] Rapaka, A., Novokhodko, A., Wunsch, D., "Intrusion detection using radial basis function network on sequences of system calls," in *Proceedings of the International Joint Conference* on Volume 3, 20-24 July 2003 Page(s):1820 - 1825 vol.3.

[17] S. A. Hofmeyr , S. Forrest , and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security* , 6 (3) : 151-180 , 1998.

[18] S. Forrest , S. A. Hofmeyr , and A. Somayaji, "A sense of self for unix processes," in *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy , Los Alamitos* , CA , 1996. Page(s):120-128

[19] Snort Homepage. http://www.snort.org/

[20] STAT Homepage. http://www.cs.ucsb.edu/~rsg/STAT/

[21] T. Garfinkel, "Traps and pitfalls: Practical problems in system call interposition based security tools," in *Proceedings of Network and Distributed Systems Security Symposium*, February 2003. Page(s):163-176

[22] Tal Garfinkel, Ben Pfaff, Mendel Rosenblum, "Ostia: A Delegating Architecture for Secure System Call Interposition," in *Proceedings of the Internet Society's 2004 Symposium on Network and Distributed System Security*.Page(s):187-201

[23] Terrance Goan-PI, "ICE: Intelligent Correlation of Evidence for Intrusion Detection," Technical Report #183, Stottler Henke Associate Inc.

[24] Tsung-Yi Tsai, Kuang-Hung Cheng, Chi-Hung Chen, Wen-Nung Tsai, "An Intrusion Prevention System using Wrapper," in in *Proceedings of International Computer Symposium* on 15-17 Dec 2004 page(s): 1218-1223

[25] Warrender, C., Forrest, S., Pearlmutter, B., "Detecting intrusions using system calls: alternative data models," in *Proceedings of the 1999 IEEE Symposium* on 9-12 May 1999 Page(s):133 – 145.

[26] Yan Qiao, Xie Weixin, "A Network IDS with low false positive rate," in *Proceedings of the 2002 Congress on Volume 2*, 12-17 May 2002 Page(s):1121 – 1126.

[27] Yasin, M.M., Awan, A.A., "A study of host-based IDS using system calls," Networking and Communication, 2004. INCC 204. International Conference on 11-13 June 2004 Page(s):36 – 41.

[28] Zhang Yanchao , Que Xirong , Wang Wendong , Cheng Shiduan , " An immunity-based model for network intrusion detection," in *Proceedings of ICII 2001 - Beijing . 2001 International Conferences* , Volume : 5 , 2001 page (s) : 24-29 vol.5.

[29] Zhao Junzhong, Huang Houkuan, "An evolving intrusion detection system based on natural immune system," in *Proceedings of 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering Volume 1*, 28-31 Oct. 2002 Page(s):129 - 132 vol.1.

[30] Zhou-Jun Xu , Ji-Zhou Sun , Xiao-Jun Wu , “An immune genetic model in rule-based state action IDS,” Machine Learning and Cybernetics , 2003 International Conference on Volume 4 , 2-5 Nov . 2003 Page (s) : 2472-2475 Vol.4.

[31] 李駿偉、田筱榮、黃世昆，入侵偵測分析方法評估與比較，Communications of the CCISA Vol. 8 No.2 March 2002. pp 21-37