第 1 章、序論

由於網際網路快速發展，電子商務應用日益普遍，於面臨全球化資訊之際，舉凡個人日常生活、企業各項營運作業及政府機關行政作業及便民服務措施，均已高度仰賴通訊與資訊系統。若通訊與應用之資訊系統無法正常運作，將使大部分的業務停頓，屆時社會將陷入恐慌及失序，為使社會永續繁榮發展，資訊安全相關議題日益受大眾廣泛討論與重視。

1.1 研究動機

現代企業面臨全球化業務推動與競爭，企業網路之需求量增加，多數之資訊系統均已連接於網際網路上，雖提供更便捷快速之業務處理工具，不幸地卻面臨各項資訊系統普遍均有弱點存在，而其弱點除可能導致電腦系統的誤用與濫用外，亦將成為網際網路上駭客與暴力分子之攻擊目標。故近幾年入侵偵測相關系統之需求增加[LCP00]，其投入的開發研究亦大幅成長。

電腦系統普遍存在安全性弱點，不論微軟 IE、UNIX 緩衝區溢位或我們使用的應用軟體與作業系統，存在不同等級的安全弱點。而傳統消除安全性問題的方法，係從軟體工程著手，藉由執行安全

明確的安全政策，來正確地製作程式與設定系統。

但以上是理論假設，實際上並無法達到上開假設，因為電腦系統並非靜態，它可能被電腦供應商、系統管理者或使用者改變，若要確認系統安全，除必須花費相當多時間外，因系統隨時可能的變動，使確認系統安全更加困難，故必須面臨接受安全政策、程式製作與系統設定存在弱點的事實，並透過工具使用例如：入侵偵測系統(Intrusion Detection Systems; 簡稱 IDS) ，以改善安全性問題。

1.2 研究範圍

入侵偵測系統是資訊安全與架構的重要一環，其主要目的是由系統的正常活動，區別可疑的入侵行為。依入侵偵測系統的製作方法可分為 Misuse 入侵偵測與 Anomaly 入侵偵測兩類。相關內容分述於下：

1. Misuse 入侵偵測：

   係指利用已知著名的攻擊及系統弱點，進行目前活動的比對，以找出是否為攻擊行為。此方法的缺點是無法預測未來可能發生的入侵。

2. Anomaly 入侵偵測:

   針對組織的系統正常活動，進行記錄並建立系統的使用特徵值

與模型，例如 CUP 與 I/O 的使用量。當目前活動超出系統正常值時，立即發出入侵警告。此方法能偵測出未來可能發生的入侵活動。

入侵偵測系統大多以特徵值辨識為基礎，Misuse 入侵偵測係事先由攻擊的模式中找出特徵值，以區別進入的封包是否為攻擊封包，而 Anomaly 入侵偵測則先找出正常行為的特徵值，再進一步區別是否與正常行為的特徵值相符，若不符則判定為入侵行為。但這些系統的執行效率均侷限於特徵值資料庫的效能（例如：資料庫大小、搜尋速度等）。

在 1987 年 Dorothy Denning [Den87]是第一個採用 Anomaly 入侵偵測的入侵偵測系統。早期有學者採用統計式 Anomaly 入侵偵測例如 IDES(Intrusion Detection Expert System)與 NIDES (Next-Generation IDES) [JV91] [AFV95]。但此種 Anomaly 入侵偵測系統運作時，於實際環境因素遭遇困難，通常系統的正常行為均會有微量的變化，若門檻值設定過低時，此種微量的變化易造成系統的誤判，而此種發生微量變化的頻率與次數於現實環境則經常發生。

為尋找網路封包的特徵值，2002 年 Hong Han[FBV2002]等人提出使用資料探勘方式尋找特徵值。為降低入侵偵測系統的誤判率，2002 年 German Florez [FBV2002]等人提出模糊資料探勘入侵偵測系統。

因多數的入侵偵測系統均建立特徵資料庫，當網路封包進入時，系統比對特徵資料庫的特徵規則，以決定該封包是否為正常或攻擊封包，在此種入侵偵測系統的效能均受限特徵資料庫的執行效率，故於 2001 年 Susan C. Lee [LH01]等人提出階層式類神經網路架構的 Anomaly 入侵偵測，以類神經網路所架構的入侵偵測系統，事先將網路封包的特徵值訓練網路(Neural-Network)，並將訓練完成的網路進行入侵偵測，故其效率必定遠高於具特徵資料庫的入侵偵測系統。

本研究範圍，將依內政部土地測量局現有網路封包及流量情形，進行網路作業行為分析、觀察網路非法使用及異常行為。

研究預期效果說明如下：

（一）運用內政部土地測量局防火牆稽核紀錄檔(Log)進行網路封

包資料分析，適時檢討防火牆規則及內容，並配合調整規則順序。

（二）依據網路分析結果，適時修正現有網路架構。

（三）蒐集相關駭客攻擊模式與行為，進一步檢討修正內政部土地測量局相關安全政策。

1.3 研究報告架構

本研究報告於第 2 章將簡介入侵偵測系統之相關研究，第 3 章將針對防火牆紀錄進行資料分析，利用前置碼基礎匹配方式(Prefix-Based Matching Scheme)將封包簡易分類，並搭配二元樹，將來源 IP(Source IP)及目的地 IP(Destination IP)依前置碼資料簡化分類，藉以適時調整及修正防火牆規則，及異常行為的偵測。第 4 章將研究結果做一個結論並研擬未來研究方向。

第 2 章、入侵偵測相關研究

所謂入侵行為，可定義為任何一連串的活動，企圖破壞整體性(Integrity)、機密性(Confidentiality)、或資源的使用者均屬之[HLMS90]。

入侵偵測系統(Intrusion Detection System，簡稱 IDS)是資訊安全與架構的重要一環，藉由連續監控在電腦系統或網路上的動態行為的特徵，以區別可疑的入侵行為。而入侵偵測系統基本上具有三個元件： 1.資料收集 2.資料分類 3.表報輸出。

一般而言，入侵偵測系統與安全分析的工具不同，例如 SATAN[FV95] 與 COPS[FS90] 被應用於掃瞄系統的弱點安全漏洞。掃瞄系統並非入侵偵測系統，因未偵測入侵者侵入的系統特徵及證據，僅掃瞄系統弱點、錯誤設定或較差的密碼。其他重要的系統以解決電腦安全問題，但非入侵偵測系統者有加密[Den92]、對於鑑別身分(Authentication)與安全的溝通[SS96]、病毒防範(Virus Protection Scheme)[Kep94]僅是靜態碼掃瞄，非動態行為特徵值。

依入侵偵測系統的製作方法不同，可區分為 Misuse 入侵偵測與 Aanomaly 入侵偵測兩類。但有些系統合併二類的優點如 NIDES, 或 Denning 入侵偵測的通用模型[Den87]。

其中 Anomaly 入侵偵測系統是假設入侵行為未知，藉由所建立正常行為的模型，以偵測系統的行為，若違反正常行為的模型則視為入侵行為。首先採用 Anomaly 入侵偵測技術建立之通用模型的入侵偵測系統，係由 Denning 提出[Den87]。Anomaly 入侵偵測有兩個基本步驟[HFS98]：第一階段建立正常行為的概述檔案(Profile)或資料庫，第二階段使用資料庫進行系統行為的偵測。

為實現 Anomaly 入侵偵測，有很多學者嘗試以不同的型態實現，有學者採用類神經網路 (Neural Networks)，以偵測於軟體上不正常的行為[GC98]也有採用資料探勘將 TCPdump 的資料記錄[BJPW2001]，甚至也有使用統計學方法以尋找異常的行為，並建立行為概述的資料庫[AFV95], [KYM02], [LJ88], [PN97], [LP99]。

Misuse 入侵偵測將入侵特徵與利用已知的入侵者行為進行掃瞄。有學者使用專家系統學習已知的入侵特徵值的資料，例如 IDES/NIDES, 或 Stalker[SW94]，將已知的入侵行為加入專家系

統的規則。另有學者設計自動產生入侵者特徵值 [IKP95]，或使用 Colored Petric Nets 進行特徵值比對[KS94] [Kum95]。

入侵偵測系統有很多不同的架構，可以集中由一台電腦負責運作，亦可分散到很多台電腦執行[HJS93] [Kum95]。但幾乎所有的入侵偵測系統均是集中，只有少數入侵偵測系統被提出來採分散多台電腦執行，例如自動代理程式方法(Autonomous Agent Approach) [CS95]。

針對網路封包或一般主機為偵測對象，入侵偵測系統又可分為主機型(Host-Based)與網路型(Network-Based)兩種，其詳細內容說明如下：

1. 主機型入侵偵測系統(Host-Base IDS)：

   此入侵偵測系統可直接與主機伺服器上之作業系統與應用程式做密切的整合，故可偵測出許多網路型入侵偵測系統所查覺不出的攻擊模式(如網頁置換、作業系統的 Kernal 竄改)。

2. 網路型入侵偵測系統(Network-Based IDS)：

   此類入侵偵測系統[HML92] [HDL90]係針對網路上的連線狀態及傳輸封包的內容進行監控。

入侵偵測系統依採用的技術，有以規則為基礎的入侵偵測系統 IM[TCL90]，或應用統計方法 NIDES[Lun93] [LTG92] [AFV95] [HLMS90]產生使用者的行為模型所建立的入侵偵測系統。也有學者研究權限行為的特徵值[CGK94]進行入侵偵測。

許多的入侵偵測系統讀取封包的內容，以觀察行為與建立使用者的概述檔(Profiles)，但隨者加密的技術應用愈加廣泛，欲單獨藉由觀察封包的內容以獲得行為資訊已不可能，故有學者提出使用於加密環境的入侵偵測系統[TA04] [Yas02]。

因入侵偵測系統應用於實際商業用途或研究理論的架構上，均需考慮設定(Configuration)與擴展(Scalability)或效率問題，故有學者使用機器學習模型(Machine-Learning Paradigm)— 類神經網路 [Lun93] [TCL90] [AFV95] 與模糊推理系統(Fuzzy Inference System)，設計入侵偵測系統。

大部分系統由稽核紀錄蒐集使用者行為的概述例如：IDES/NIDES [Lun93] [TCL90] [AFV95], Wisdom&Sense[LV92] and TIM [TCL90])並進行偵測，而即時型入侵偵測系統則製作上較困難。

2.1 應用模糊資料探勘於入侵偵測系統

於 2002 年 German Florez [FBV2002] 等學者應用模糊資料探勘技巧，找出代表企業或組織的正常行為模式的入侵偵測系統。為提高正確性與效率，本篇採用模糊關聯法則(Fuzzy Association Rule)，將網路上的稽核資料進行資料探勘以得到正常行為的模型。建立正常行為模型後，即進入偵測異常行為階段，此時由新的稽核資料產生糊關聯法則後，再與正常行為作相似性比較，若相似性的值低於門檻值時，則表示可能為入侵行為，故需發出警告或直接阻擋。

本篇除採用模糊資料探勘外，尚運用 Borgelt's Prefix Trees 為基礎計算糊關聯法則的支持度與信心度，並以基因演算法(Genetic Algorithms)進行特徵值的選擇與最佳化。

利用對稱相似函數(Symmetric Similarity Function)如公式(1)，進行目前活動與正常活動的比對。兩個關聯規則 R1, R2 定義如下：

$$similarity(R_1, R_2) = \begin{cases} 0 & if (X \neq Y) or (X' \neq Y') \\ \max\left(0.1 - \max\left(\frac{|c - c'|}{\max(c, c')}, \frac{|s - s'|}{\max(s, s')}\right)\right) & if (X = Y) and (X' = Y') \end{cases} \quad (1)$$

R1：X → Y, c, s,

R2：X' → Y', c', s'

其中 X, Y, X', Y'均是模糊項目集。

R1 與 R2 的相似程度為 similarity(R1, R2)。

利用相似度公式 similarity(R1, R2)，計算 S1, S2 兩個規則集合的相似程度如公式(2)。

總相似程度如公式(3)。

$$s = \sum_{\substack{\forall R_1 \in S_1 \\ \forall R_2 \in S_2}} similarity\ (R_1, R_2) \tag{2}$$

$$similarity\ (S_1, S_2) = \frac{s}{|S_1|} * \frac{s}{|S_2|} \tag{3}$$

為了減少 Apriori 演算法的執行時間，作者提出修正關聯規則最小支持度與信心度的計算如公式(4)、(5)。

$$NewMinSup = (1 - \min\_sup)\frac{(NumberFeatures - CurrentFeatures)}{NumberFeatures} K + \min\_sup \tag{4}$$

$$NewMinConf = (1 - \min\_conf)\frac{(NumberFeatures - CurrentFeatures)}{NumberFeatures} K + \min\_conf \tag{5}$$

修正的公式中，加入關連規則的左邊與右邊的特徵值的項目數量的影響，利用門檻值 K 控制產生的關聯規則的數量，當 K 值越大，其產生的關聯規則越少；反之 K 值越小，則產生之關連規則越多。另外，考量特徵值越多的情形下，所涵蓋的資訊量越多，故當關聯規則的特徵值的項目越多，則新的最小支持度與最小信心度越小，故產生的關聯規則越多。舉例而言，如下圖中，有一個關聯規則，有 3 個特徵值項目分別為 nrf, npf 與 tsize。

$$(nrf=low)\,(npf=high) \rightarrow (tsize=medium)$$
$$support = 0.346, confidence = 0.82$$

如果 min_sup = 0.2 與 min_conf = 0.8 則此項關聯規則將會被採用。若採用 K = 0.5 則新的支持度為 0.49 與信心度為 0.872，因門檻值提高提高，故此關聯規則將會被捨棄。

$$NewMinSup = (1 - 0.2)\frac{(11 - 3)}{11}(0.5) + 0.2 = 0.49$$

$$NewMinConf = (1 - 0.8)\frac{(11 - 3)}{11}(0.5) + 0.8 = 0.872$$

於 1998 年 C. Kuok[KW98]等人提出模糊規則在資料庫的探勘。因布林型關聯規則的資料探勘(例如購買奶油與牛奶則也會購買麵包)，在過去已有相當多的專家學者，提出相關的論文。但是現實的資料庫中，存在許多的類別型 (Quantitative Attributes)的資料，例如：Integer, Categorical, Numerical 的屬性資料，若直接利用布林型關聯規則的資料探勘，則必須進行轉換。如果利用模糊集合來處理類別型的資料，因可使用人類慣用的語詞，故將更易了解；並且運用於數值資料的切割，將避免界限銳利(Sharp Boundaries)的問題。

所謂模糊關聯規則其型式如下：

If X is A then Y is B

其中 X 與 Y 是屬性集合，A 與 B 是用來描述 X 與 Y 的的模糊集合。

例如：X = {輸入密碼錯誤次數, 掃瞄主機 Port }，

Y = {密碼猜測攻擊}，A,B = {高，中，低}，

IF 輸入密碼錯誤次數 is 高 then 密碼猜測攻擊 is 高。

為因應網路上有突然暴增之流量情形，其需產生的關聯規則大增，使得原來 Apriori 演算法執行時間暴增，故本篇提出新的方法，可動態調整關聯規則的門檻值，減少關聯規則的數量，以增加系統的效能與提高正確性。

2.2 應用 Metadata 於 Anomaly-Based 入侵偵測系統

為建立使用者行為的概述，於網路或主機系統上，以讀取封包內容來建立。但隨著加密的技術的進步，越來越少的封包內容可供分析，而為完成在加密環境的入侵偵測，本篇藉由蒐集與分析使用者通訊協定與通訊期間的活動(Session Activity)，建立各種型態的 Metadata。

本篇使用的 Anomaly-Based 方法稱為 BSEADS(Behavioral Secure Enclave Attack Detection System)。其所模擬的網路

環境稱為 SEADS(Secure Enclave Attack Detection System)，係由佛羅里達州立大學(Florida State University)的 SAIT (Security and Assurance in Information Technology Laboratory)所建立。

2.2.1 入侵偵測模型簡介

SEADS 其主要運用於加密流量的入侵偵測[Yas02]，依其應用可分為知識型 SEADS(Knowledge-Based SEADS 簡稱為 KSEADS)與行為型 SEADS(Behavioral-Based SEADS 簡稱為 BSEADS) 兩類。KSEADS 是 Misuse 偵測方法，其主要係將流量與已存在的攻擊特徵值進行比對，故其缺點為新產生之攻擊行為無法偵測到，因其特徵值不存在於該攻擊特徵資料庫內。BSEADS 則係採用 Anomaly 入侵偵測方法，其技術係用統計相關的技術將過往歷史的活動，由入侵偵測系統產生各種「正常活動概述」，當不同於「正常活動概述」的異常行為發生時，即可偵測到入侵行為。

2.2.2 行為分類

本篇文章將行為概分為三類，正常行為、異常行為(Abnormal Behavior)與惡意行為(Malicious Behavior)。正常、異常與惡意行為其程度很難藉由明確的界線加以控制。例如使用者是正常的惡意行為，即使用者於執行一些例行的活動時，卻加入屬於惡意的活動，如此 Anomaly 偵測系統無法偵測到此攻擊。另外，人類正常行為之自然變化，易被視為異常行為。本篇將行為分類如下圖：

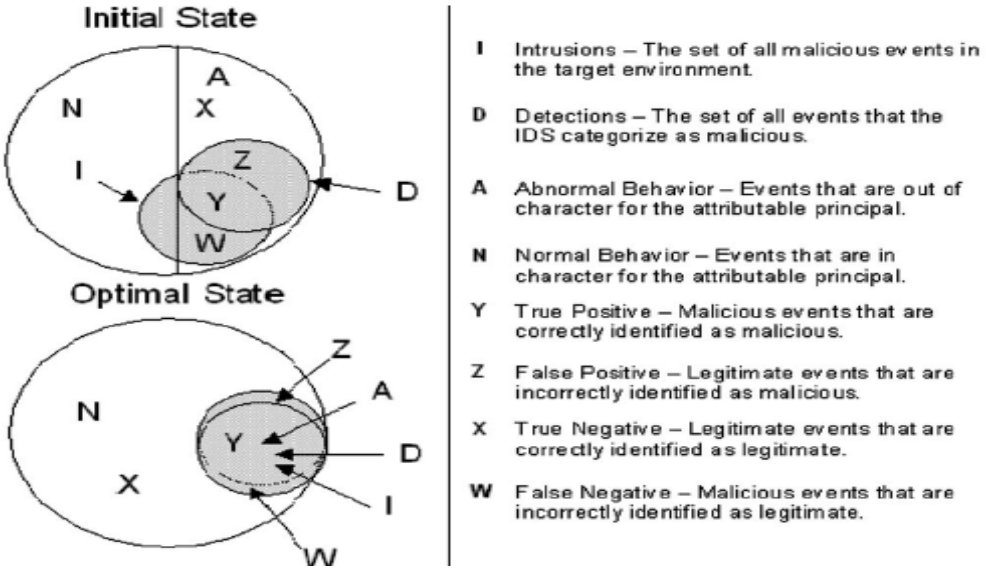


圖 2.1 正常、異常與惡意事件

圖中 I 表示實際的入侵行為，其涵蓋有 W 與 Y 聯集，Y 表

示惡意的行為被正確地識別，而 W 表示惡意的行為被誤判為合法行為。另外，D 表示入侵偵測系統所識別為惡意的攻擊行為，其涵蓋 Y 與 Z 聯集，Z 表示合法的行為被誤判為惡意行為。於最佳情況下，W 與 Z 的誤判率應越小越好。

2.2.3 以時間概念進行正常行為分析

使用者的活動與時間(小時、天、週、季節、假日、休假日等)變化有關。為避免高的合法活動越軌為不正常行為，行為型入侵偵測系統必須識別使用者隨者環境與時間變化之嗜好改變。

本篇將時間一天 24 小時分為六類，Early Morning, Morning, Afternoon, Evening, Night, 與 Entire Day。行為入侵偵測引擎(Behavioral Intrusion Detection Engine，簡稱 B-IDE) 在每日預先定義的時間上，執行系統行為的檢查。每一類於預先定義的時間區段，藉由分析群體的活動(Session

Activity)，將可詳細地偵測行為的改變，以找出越軌的行為。

2.2.4 BSEADS 機制

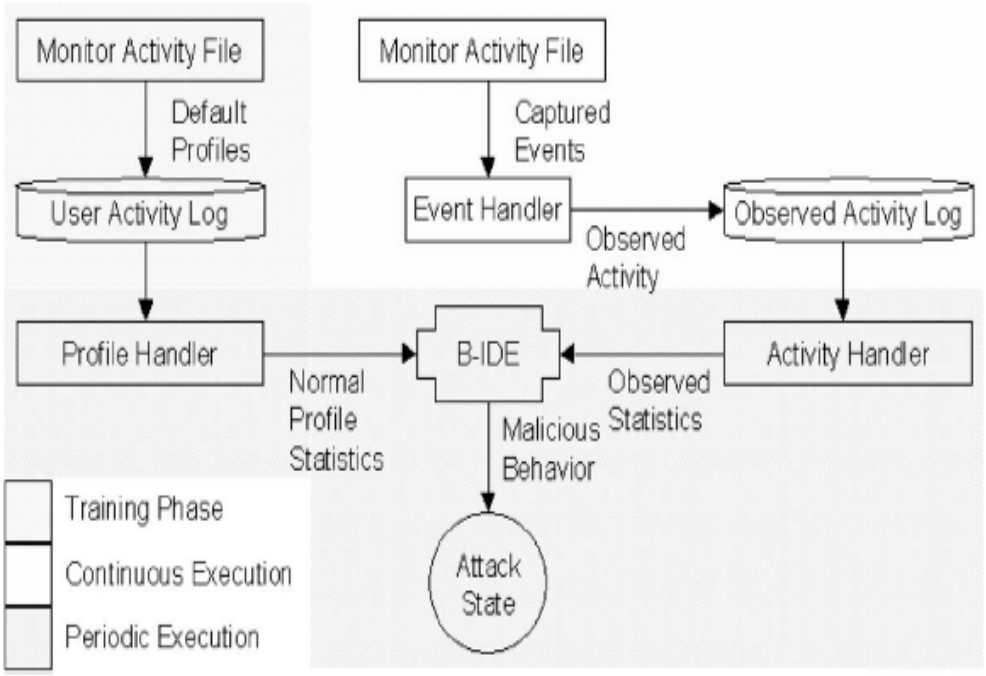


圖 2.2 BSEADS 機制

在加密安全協議環境，有很多針對安全協議分析與驗證的研究[HK01], [Low96], [Syv94], [WS96], [WL92]。

BSEADS 建立使用者活動紀錄，係由監督活動檔中萃取出使用者無攻擊行為的資料。這些訓練的處理輸出正常使用者行為的概述

檔案。

一個事件管理器(Event Handler，簡稱 EH)，藉由監督器動態地萃取正進行中的協議活動(Protocol Session Activity)。行為入侵偵測引擎係用於每一行為之分析(Behavioral Analyzer; 簡稱 BA)。

活動管理器(Activity Handler, 簡稱 AH) 以維護每一個電腦(Principal)的觀察活動紀錄且在這些紀錄上執行一些方法，並且供 B-IDE 要求呼叫。PH(Profile Handler)維護每一個電腦使用者活動的紀錄。

2.2.5 使用者活動

使用者的安全協議活動(Security Protocol Session Activity)儲存為紀錄檔，以作為使用者的基本概述。在使用者紀錄檔中，暫時的 Metadata 資訊包含時間開始與結束，使用年、月、日、分、秒的格式。

2.2.5.1 正常使用者的概述

使用者的概述包含虛擬(Synthetic)與實際方法(Real Methods)訓練[LS98]。

使用人工方式準備資料，虛擬技術藉由執行應用系統，以產生各種正常模式的行為，於訓練階段可以保證所產生的描述檔為無入侵行為紀錄。

2.2.5.2 使用者活動紀錄

使用者活動紀錄 (User Active Log； 簡稱 UAL) 主要儲存歷史資料，是 MetaData 的統計表示，以使用者 Session 活動為基礎。在系統的每一個使用者均有相對應的紀錄，以包含所有使用者執行的所有活動。

資料庫的格式：

1. 每一會談(Session)的唯一描述(例如：開始時間與結束時間)。

2. PR 欄位表示相對應的會談。

3. 會談結束(Session Completion) 定義為 C，包含反應會談

是成功或失敗。

4. 金鑰加密的長度(Key Strength,簡稱 KS)，用以描述加密的強度。

5. 會談的識別(Session Identifier; 簡稱 SID)，是每個會談被儲存的一個唯一值(Globally Unique Identifier; GUID)。

2.2.5.2.1 時間衰退(Time Degradation)

實際上人類行為是動態的，正常行為似乎總是在改變，藉由限制使用者靜態概述的生命週期，過去行為隨著時間增加，退化率愈高。時間差計算如下表：

User Activity Log	Degradation Factor		
n	0%	0%	0%
n to 7 days old	40%	20%	10%
(n – 7) to 14 days old	60%	30%	15%
(n – 14) to 21 days old	80%	40%	20%
(n – 21) to 28 days old	100%	50%	25%
Weekly Standard Deviation	**0 = Low**	**> 10 = Moderate**	**> 20 = High**

圖 2.3 時間退化計算(n = 最近觀察日期)

時間退化演算法(Time Degradation Algorithm) 係執行於使用者活動紀錄(User Active Log，簡稱 UAL)，其執行結果為正常

概述統計。

以 4 週為例，時間愈久其退化率愈高。另外，週的標準差愈高，表示其活動變動性愈高，其退化率就越低，例如週標準差大於 20，此為高變動性(High Volatility)，活動更明顯，故其活動被降級速度慢。

2.2.5.2.2 正常概述統計(Normal Profile Statistics)

透過資料與萃取矩陣資訊(Retrieve Metric Information)以重新組合正常使用者活動。計算每一個行為分析矩陣值，並將結果的資料集合回傳。

2.2.5.2.3 使用者觀察概述

當事件管理器(Event Handler) 從監督器萃取 session 活動時，會立即加入其相對應的使用者觀察概述檔。

一段期間後，B-IDE 向活動管理器(Activity Handle) 要求每一使用者的觀察統計，並執行標準差測試是否有任何變動或集

中行為，也比較正常概述的統計的集合，以偵測不正常的行為。

2.2.6 行為分析器

行為分析器的基本概念是偵測行為特徵(以偵測惡意的攻擊者，其方法採用監控 Metadata 方法，將監控的活動分為數模組稱為行為分析器 BA，分別針對平行會談(Parallel Session),失敗會談(Failed Session),重送會談(Replay Session)與弱點會談(Weak Session)等弱點行為進行分析，如下圖：

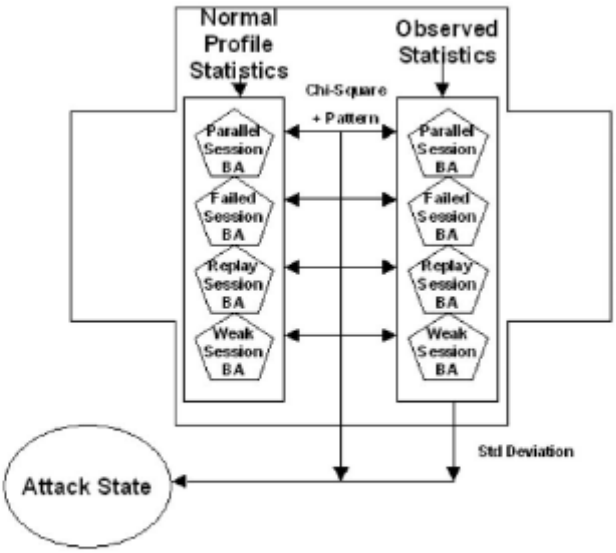


圖 2.4 行為入侵偵測引擎

本篇發展 BA 是結合動態矩陣，此動態矩陣係由靜態表所萃取

出來。每一個 BA 使用唯一的矩陣集合，以描述特殊的 BA 其目標功能與目的。本篇 BAs 由觀察統計與正常概述統計產生。兩個集合採用 Chi-Square 統計，以決定差異值，並採用特徵識別值，比較偵測觀察活動的不正常特徵。

2.2.6.1 平行會談(Parallel Session)

平行會談攻擊表示兩個或更多的會談同時被執行，攻擊者可以收集另一個正執行中的會談資訊進行攻擊[WL92]。平行會談行為分析器將被使用來監控這形態的活動。

2.2.6.2 失敗會談(Failed Session)

入侵者於攻擊系統時不需要成功，僅須藉由多個會談失敗，造成很多無意義的資訊即可干擾系統的運作。例如暴力攻擊(Brute Force Attack)，攻擊者使用嘗試錯誤(Trial-and-Error)技巧，嘗試所有合法值的各種可能組合，有順序的測試， 進行解開加密的資料，例如：密碼或 DES 金鑰(Data Encryption Standard)。因暴力攻擊會產生很多失敗會

談，故失敗會談行為分析器 即可用來偵測此種活動。

2.2.6.3 弱點會談(Weak Session)

現代加密安全協議一般是允許協商，以建立加密參數例如金鑰長度，當限制金鑰大小時，將增加暴力攻擊的機率，若限制金鑰輸入的次數則可以減少暴力攻擊。

當使用者採用有弱點的加密參數時,行為分析器 BA 可以比對 Metadata 的行為特徵值及其加密參數。

另一種利用加密的弱點格式所衍生的攻擊，即加密組的回復攻擊(Rollback Attack)，當主動攻擊者為惡意破壞加密的型式，而編修加密組的 Cleartext List，攻擊者即可強迫使用者用 Export-Weakended 加密[WS96]。上項之攻擊可使用弱點會談行為分析器(Weak Session Behavioral Analyzer)偵測。

2.2.6.4 重送攻擊(Replay Session)

在重送攻擊中，駭客分析網路封包以監控並拷貝網路封包，當收集到必要的封包時，即可萃取必要的封包資料例如：

數位簽章鑑別資訊與記錄溝通訊息後，可進行重送攻擊。

為預防重送攻擊每一會談加入時戳，形成唯一的識別。但不幸地，使用此種機制仍不能保證安全，為預防重送攻擊須增加協議的確認機制(Protocol Verification)。為偵測上項攻擊可用重送攻擊行為分析器進行偵測。

2.2.7 偵測方法

本篇偵測方法係應用於 BAEADS，其行為分析器主要元件有三類，說明如下：

(1)衡量分散度(Spread)：每一個非標準化的行為分析器採用標準差進行標準化，以觀察行為的分散程度。

(2)衡量正常度(Normality)：比較觀察統計與歷史的正常概述統計，其統計方法採用 Chi-Square 測試。

(3) 衡量特徵值比對(Pattern)：使用特徵識別進行觀察特徵與歷史特徵相比較。

2.2.7.1 分散程度衡量

分散度是衡量 Session 活動分布在時間的某一區段或整個 24 小時。進一步地考慮時間區域性(Time Locality) 與頻率，以決定是否為攻擊。藉由分析觀察統計，我們能夠依其分散的特性偵測到攻擊行為。因為攻擊行為通常具有群化與高集中的特性。Lee 與 Stolfo 提出演算法以偵測系統在某個時間類別上的不正常活動，但此方法未衡量活動的分散特性[LS00]。

針對每一個於觀察統計的行為分析器執行標準差測試，如果標準差的值大於 1 即提出警告可能為駭客入侵行為。

2.2.7.2 正常行為衡量

所謂 BSEADS 正常行為包含正常概述統計，其內容係由訓練歷史資料而動態產生，並將使用者行為分類為惡意(Malicious)或非惡意兩類。利用 Chi-square 測試，檢查是否為正常行為，其公式如下

$$X^2 = \sum_{i=1}^{n} \frac{(X_i - E_i)^2}{E_i}$$

首先，設定信心度及接受或棄卻的假設區域，其假設為正

常行為或異常行為，其分析內仍包含觀察與正常概述統計。本篇使用 Chi-Square[YC01]區別兩個或更多矩陣的相關性，在程式執行期間，Chi-square 均會被執行。這些矩陣是 BA 於多個時間點執行計算的值。

由過去的正常概述計算上限值，其公式如下：

控制上限(Upper Control Limit) = $\overline{X}$ + 2S。

Chi-square 測試是對每一個 BA 於某個時間類別的正規化計算。如果大於上限值，則發出有不正常的活動。

2.2.7.3 特徵值衡量

入侵偵測活動通常是個別活動的加總，而其個別的活動並非異常，目前並沒有任何機制可以識別惡意與開始異常的微小差異。

採用標準差統計活動的衡量，可有助於偵測不正常行為的尖峰時段，但這些衡量方式對於事件發生的順序及相互關聯沒有偵測能力。另外，面臨的困難是如何決定正確的入侵行為門檻值。

預先特徵認知是假設有事件發生有順序性。

E1 -> E2 -> E3 -> (E4 = 90%, E5 = 10%)

如果事件發生的順序是 E1、E2、E3 後，再發生 E4 的機率是 90%，發生 E5 的機率是 10%。因為要完全符合左邊的規則，即有很強的順序性，故其缺點是無法掌控使用者的行為的變異。

使用 Anomaly-Based 方法另加上統計衡量指標，系統可以監控事件預先指定特徵的發生情形，這些特徵被儲存於使用者活動紀錄。因此，觀察活動會以事件的順序去比較正常特徵值的活動，並從使用的特徵值差異產生攻擊的資料特徵概述檔。

Time (t)	1	5	6	7	11	14	19	23	25
Event	a	c	b	c	a	c	a	b	c

圖 2.5 時間邊界特徵比對

假設事件 a, b, c 可以表示 SSH 會談(SSH Session)失敗，採用 “follows” 語意，決定特徵發生最小的時間，開始有 a 接下來發現 b，接下來發生 c 的機率。以概略檔特徵(Profiled Pattern)識別的發現，在 10 次的單位的匹對。abc 最小時間發

生機率，t = 1 是 6，t = 11 是 25-11 = 14。因此，t = 11，這是異常, 他經過 10 個時間單位完成。我們計算每一個 a 值，其時間最差的情形 $O(n^2)$。

本篇採用離散概約比對(Descrete Approximate Matching)為基礎的特徵值比對演算法(Pattern-Matching Algorithm)。LCS(The Longest Common Subsequence)問題[WF74]，如果事件輸入字串 abcadacacbac，概寫檔的特徵值為 adab。

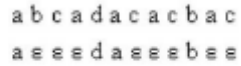


圖 2.6 概約比對方法

概約比對方法(Approximate Match Approach)，如上圖ε輸入與特徵值不符，LCS 可以使用動態程式[KS94]解決，其時間複雜度為 O(mn)，m 是特徵值長度，n 是輸入。當特徵值長度足夠小，可以與電腦的字長度相符，則匹對作業將可執行，執行的時間複雜度為線性時間[BG89][WM91]。例如在事件 a 直到 t4 有兩個前面的事件, b 在 t2 與 c 在 t3。

指定限制 Tt4 - Tt2 ∈ [0, 5]

與 Tt4 - Tt3 ∈ [5, 10]

加強限制在最後事件相對於最後的總時間內，其特徵值必須相符。限制的區間由過去特徵識別的歷史描寫檔(Histrory Profile)決定。指定 Tt4 – Tt3 ∈ [5, 10]，事件在 t3 必須在事件 t4 之前 5 個時間單位出現，但不可在前 10 個時間單位出現。

2.2.8 BASEADS 實例

本篇使用的模擬環境 Windows 2000 平台，標準 C++ 使用 Standard Template Library(STL)，用以發展一個有效率於加密環境的入侵偵測系統。目前加密環境越來越多於網際網路上的流量，很多傳統的 IDS 研究[DS99][LP99] [HFS98]，與現在的架構接近。

本篇在實驗中使用兩個監測活動檔，第一個在起始訓練階段，以建立無攻擊活動的紀錄。BSEADS 的第一個活動建構 UAL 由動態的訓練監測活動檔，共收集 4 週的資料。

開始的模擬時間 12:00 A.M.，由 EH 要求監測活動。模擬在即時環境抓取通訊協定會談。這個會談傳送到 EH，事件以每

一個電腦(Principal)為基礎解析，以傳送予累積 Metadata 放入每一個電腦觀察活動紀錄檔。

時間類別為黎明 Early Morning(EM),結束是 6 小時，入侵偵測系統依照順序檢查行為分析。於觀察統計尋找分布在 EM 類別的類別，使用觀察描寫統計與正常描寫統計應用 Chi-Square 統計檢查是否為正常。

觀察特徵與歷史特徵相比較，並檢查其他時間分類(Morning, Afternoon, Evening, Night)活動。一旦晚間時間類別完成，將檢查其分散度、行為，與執行整日累加的特徵識別。

BSEADS 由檢查分散度，以群體為總合(所有時間類別的總合)檢查一致性(Uiformity)。系統接下來以個別小時檢查系統的一致性與特徵分析。最後，以整日的活動為基礎進行測試。

2.2.8.1 平行會談行為分析器(Parallel Session Behavioral Analyzer)

活動觀察係 EH 由監控活動檔找出電腦 A 與 B，是否存在中間人攻擊(Main-in-the-Middle Attack)。

活動觀察由 EH 針對每一台電腦放在活動觀察記錄檔。一旦中午類別實現，B-IDE 檢查整個下午累積的行為。要求從概略檔萃取器(Profile Handler;簡稱 PH) 得到正常概略檔統計。

下午平行會談 BA 之正常概略檔統計被查詢，B-IDE 從 Activity Handler(HA) 要求觀察統計。 BSEADS 使用標準差衡量觀察統計的分散程度(Spread)。計算的結果為 1.304。表示此事件在此段時間沒有常態分佈。

B-IDE 將觀察與正常概略檔統計檢查，其計算結果是 2，BSEADS 獲得控制上限 1.50，因為 Chi-Square Value 大於控制上限為異常行為。

下一階段檢查特徵識別方法。平行會談活動的觀察特徵在中午時段分為 0 到 100。

Time (t)	1	4	12	14	58	65	67	74	100
Event	SSL	SSH	IPSec	S/MIME	SSL	SSH	SSL	IPSec	SSH

圖 2.7 觀察特徵之平行會談活動中午時間類別

歷史特徵(Historical Pattern) SSH, IPSec, SSL ，SSH 被萃取與計算 50 時間單位。特徵值發生的機率最小時間，在 Observed Data t = 4 是 61 以上。因此 t = 4 超過 50 個時間單位，將被視為不正常，BSEADS 因此會發出攻擊訊號。

2.2.8.2 失敗會談行為分析器(Failed Session Behavioral Analyzer)

失敗會談分析器活動，在晚間時間類別，M 與 B 間有多列的失敗會談。

在晚間類別結束，B-IDE 執行一個系統檢查行為。PH(Profile Handler)回傳正常概略檔統計與 AH(activity Handler) 回傳觀察統計。

晚間失敗會談行為分析器正常與使用者 M 的觀察概略檔統

計，衡量觀察統計的分散度，使用標準差的結果 2.3，即發出警告。

B-IDE 分析行為的特質係以觀察與正常概略檔統計為基礎。Chi-Square 計算結果 9。獲得控制上限為是 1.49。因為 Chi-Square 的結果大於控制上限所以觀察值有不正常行為。歷史特徵值[SSH, SSH]被計算，$T_{t2}-T_{t1} \in [3, 15]$，$t_1$ 至少在 3 個時間單位，才會出現 t2，但間隔時間不可多於 15 個時間單位，以上情形可視為異常，而發出攻擊的警告。

2.2.8.3 弱點會談行為分析器(Weak Session Behavioral Analyzer)

入侵者 M 強迫 A 與 B 使用有弱點機密。觀察活動由 EH 被記錄在每一個電腦的觀察活動紀錄。

於資料累積期間，B-IDE 被要求由 PH(Profile Handler) 修改正常概略統計與 AH(Active Handler) 修改觀察統計。B-IDE 使用標準差衡量觀察統計的分散度，其計算的值是 1.30，

統計分類超過這一段期間的常態。以觀察統計與正常概略檔統計為基礎，B-IDE 隨後檢查不正常的特性，計算結果為 2，其控制上限是 1.50。經過測定有不正常行為，如同 Chi-Square 結果大於控制上限。

[SSL,SSL,SSH] 的歷史特徵,被計算大約 30 個時間單位，特徵值最小的發生的時間，在觀察資料 t = 7 時是 35，已超過正軌 30 個時間單位。

Metric	Normal	Observed
# Failed Sessions Hour 17	0	0
# Failed Sessions Hour 18	1	4
# Failed Sessions Hour 19	0	0
Total	**1**	**4**

圖 2.8 正常概略檔統計

2.2.8.4 重送會談行為分析器(Replay Session Behavioral Analyzer)

計算衡量分散度是 1.34，B-IDE 檢查不正常的特性，其結果計算是 0.5。

獲得控制上限是 1.50， Chi-Square 結果已經少於控制上

限，故沒有攻擊行為。

2.3 使用系統呼叫的順序偵測入侵

由於 Unix 的優先處理(Privileged Processes)出現安全性的問題，即其權限控管的設定太過初略: 系統管理者將系統的資源授權給使用者後，所允許的授權卻比他們實際執行所需的權限更大 [CGK94]。因此他們允許系統資源的權限大於他們要執行的權限。

因權限程序(Privileged Process)於權限處理上出現弱點，攻擊者將可利用其弱點而進入管理者的狀態(Superuser Status)。藉由監測程序之執行狀況，可有效監控使用者的行為[AFV957] [Den87] [HJS93] [LTG92] [TCL90]。

本篇權限程式的行為以不規則方式偵測，視程式為一個黑箱，即不須知道程式在網際網路上執行的功能與角色，以系統呼叫為主，以存取 Unix 的系統資源，即採用短順序的系統呼叫(Short Sequences of System Call) 當做觀察對象[FHS96]。

其靈感來自天然的免疫系統(Natural Immune Systems),因免疫系統與電腦安全[FHS96] 面臨的問題相似，系統必須保護較重要的系統被有害的代理程式傷害，故它必須能區別正常或異常行為。在免疫系統上，將區別自己“self”(對身體沒有傷害)與非自己“nonself”(危險病菌與其他外來物質)兩類。在 UNIX 程式中，由正在執行的程序所產生的短順序系統呼叫，可以提供最佳區別正常與異常行為的特性。正常行為可使用兩種方式被蒐集：(1)虛構的資料： 由應用程式產生虛構的正常行為資料。(2)真實的資料：利用程式追蹤使用者在現實環境中實際產生的資料。

2.3.1 正常行為的概述

藉由特別的程序掃描追蹤系統呼叫,建立長度為 k 的唯一順序的資料庫。一旦 Stable Database 建構完成，此資料庫將可用來偵測正進行中的行為。

實例：以下是追蹤系統呼叫

Open, read, mmap, mmap, open, read, mmap

採用 size k 的 slide window，如果 k = 3，其唯一的順序為

Open, read, mmap

Read, mmap, mmap

Mmap, mmap, open

Mmap, open, read

為效率這些順序被存在樹(tree)，樹的根(root of tree)是特別的系統呼叫。

2.3.2 衡量異常行為

正常行為的資料庫建立後，使用相同的方法建立資料庫，檢查新的行為，與正常資料庫相比較，若順序不同則稱為 Mismatches，藉由記錄多少次的 Mismatch，可以決定異常訊號的強度。多少次的 Mismatch 發生，可以決定是否為惡意行為。

為區別正常(Normal)與合法(Legal)行為，在理想的情

況下，正常資料庫包含所有正常行為的變化。我們想要不只偵測入侵，對於非正常情形的系統問題。例如：當程序執行超過磁碟空間，程序將執行一些錯誤碼，此為非正常的執行順序，而被視為正常，雖然此應該是合法行為。

如果正常資料庫包含所有正常行為的所有變化，當我們遇到其順序不在正常資料庫裡面，則被視為異常行為。本篇採用 Hamming Distance 當作順序的衡量，兩個不同順序 i 與 j Hamming Distace d(i, j)。每一個新的順序 i，其 minimal Hamming Distance $d_{min}(i)$，與正常行為的順序：

$d_{min}(i) = \min\{d(i, j) \ \forall \ \text{Normal Sequences } j\}$

$d_{min}(i)$ 表示異常訊號的強度。例如：

oprn, read, mmap, mmap, open, read, mmap

產生的 Normal Database 包含:

Open, read, mmap

Read, mmap, mmap

mmap, mmap, open

mmap, open, read

如果我們追蹤一個呼叫其順序為

Open, read, mmap, mmap, open, mmap, mmap

則讀取的呼叫其新的順序為

mmap, open, mmap

Open, mmap, mmap

因有兩個未相符(Mismatches)即兩個 $d_{min}(i)$ 值為 1。有三種衡量的時間複雜度,新順序是未相符(Mismatch)需要 k-1 次比較，因正常順序存在樹林(Forest of Tree)，其根存者不同的系統呼叫。

2.3.3 錯誤分類

IDS 以衡量觀察值為基礎，若以兩種類別判定正常或不正常，此種簡單的分類易造成 False Positives 與 False Negatives。所謂 False Positive，係指當一個順序產生是合法行為但卻被歸類為異常行為稱之；當入侵

行為未產生任何順序稱為 False Negative 亦即入侵行為的順序包含在正常行為資料庫中。在統計式的理論中，False Negative 與 False Positive 被稱為 Type I 與 Type II Error。我們衡量 Anomaly 的強度使用 d_{min}，追蹤時以 Maximum d_{min} Value 表示，計算異常值 S_A:

$$S_A = \max\{ d_{min}(i) \ \forall \text{ new sequences } i\}$$

S_A 值與順序長度 k 正規劃，比較 S_A 不同的 k 值

$$\hat{S}_A = S_A / k$$

我們想要將錯誤最小化，即錯誤的容忍率 False Negatives 多於 False Positives。False Negatives 可以被降低經由增加防禦層(Defense Layer)，但不會增加 False Positive Rate。考慮一個系統有 L 個防禦層被穿透，入侵者避開偵測的機率為 Pn（即 Pn 是 False Negative Rate)。如果每一層的偵測機率為獨立，則入侵者侵入未被發現的機率為 P_n^L。所有 False Negative Rate 因為指數計算被簡化。假設每一個層有 False

Positive 的機率 P_f，其 P_f 的期望數(Expected Number of False Positive) $P_f L$。

當在實際的環境中搜集正常行為，False Positive 可以被衡量。假如我們憑經驗蒐集正常行為，很少發生將未完整正常行為納入資料庫，亦即如果正常行為是未完整，False Positive 將會發生。可接受的順序因未包含在正常資料庫所以造成誤判。為限制 False Positives，在 $d_{min}(i)$值上設定門檻，如果任何順序 i, $d_{min}(i) \geq C$; $1 \leqq C \leqq k$, 將會被視為異常行為。長度 k 的順序 i 與所有的正常順序有相當的不同，則此順序將被標示為異常。

2.3.4 在假造環境下行為(Behavior in a Synthetic Environment)

有兩種方法選擇正常行為以定義正常資料庫：(1)產生假造(Synthetic)的正常行為，由程式產生以追蹤其行為。(2)產生實際(Real Normal)正常行為，藉由追蹤在真實正常行為的使用

者環境中獲得。假造的正常行為係對於重複結果與比較不同設定的效能與其他類別的控制實驗情況時是有用的。假造的正常行為於蒐集與評估時有一些問題。

2.3.4.1 建立一個假造正常行為資料庫(Building a Synthetic Normal Database)

本研究針對 UNIX 上不同的程序: Sendmail, Lpr 與 Wu.ftpd(執行在 SunOS 4.1.x 與 Linux)。考慮在訊息長度、訊息個數、訊息內容 ex: text, binary, encoded, encrupted, 訊息主旨(Message Subject Line)，誰送郵件，誰收郵件,Mailers的變動。另外,考慮郵件轉寄(Forwarding)的影響，郵件退回及在佇列(Queue)裡等的因素。考慮以上所有的變動影響，以及遠端傳送(Remote Delivery)與當地傳送(Local Delivery)兩種。

使用 112 種假造的訊息進行 Sendmail 的實驗，產生追蹤結合長度超過 1.5 百萬的系統呼叫。

表 2.1 Sendmail 型態的訊息數量

Type of Behavior	# of Mail Messages
message length	12
number of messages	70
message content	6
subject	2
sender/receiver	4
different mailers	4
forwarding	4
bounced mail	4
queuing	4
vacation	2
Total	112

備註：假造 sendmail 各種型態的訊息數量,每一數字表示變化的數量 。例如，12 種不同的訊息長度。

上表每一種型態的訊息已產生正常資料庫。測試 12 種不同長度的訊息，其長度由一行到 300,000 bytes。選擇最短的長度產生系統呼叫(50,000 bytes)的各種變化的特徵值，再以此為測試訊息比對。在 Sendmail 的訊息數量，首先送一個訊息並追蹤 Sendmail，再送 5 個訊息並追蹤 sendmail，依此類推到達 20 個訊息，用以測試 Sendmail 對於訊息暴增時的反應。測試訊息的內容，由送訊息的內容包含 ASCII Text、未編碼資料(Unencode Data)、壓縮資料與 pgp 加密檔案。有多少個變化被測試，與在所產生的系統呼叫特徵值中，哪一個被選。這些訊息組成正常行為的主體。將這些標

準訊息在不同的作業系統與嘗試 senmail.cf(Sendmail Configuration File)的各種變化，因此產生正常資料庫以有容忍度正確地操作情形。

依照特性的考量，以下很少或沒有影響的因素有：訊息數量、訊息內容、主旨行、誰送信、誰接收信、郵件程式、佇列。訊息長度有考慮在系統呼叫的順序的各種影響，訊息來源：遠端郵件(Remote Mail)產生系統呼叫的追蹤，與訊息長度成比例；當地郵件(Local Mail)產生的追蹤大約有相同的長度。但是當訊息長度增加時，系統訊息呼叫的順序將會被改變。轉寄郵件在遠端追蹤(Remote Trace)其影響無足輕重，在當地追蹤(Local Trace)也影響很小。退信(bounced mail)在遠端追蹤有影響，於當地追蹤仍有明顯的影響。

在每一次的測試，對於每一個不同的 k 值以產生資料庫，每一個測試有三個程序被測試即 sendmail, lpr 與 ftpd。k = 10 的結果表示如下：

表 2.2 Sendmail、lpr 與 ftpd 正常資料庫

process	Database Size N
sendmail	1318
lpr	198
ftpd	1017

備註：資料庫大小為 N，次序長度為 10

選擇的順序長度被兩個標準所決定，而此兩個標準相衝突。一方面我們希望順序長度盡可能地短，使資料庫盡可能地小與計算包含在偵測裡。另一方面來說，如果順序長度太小將無法決定正常與異常行為。所以本篇憑經驗的觀察選擇為 10。

資料庫盡可能地緊密，例如 Sendmail 資料庫只包含 1318 唯一的順序長度為 10，在製作上需要 9085 bytes。Sendmail 其中最複雜的權限程序(Privileged Processes)在 UNIX 系統上，如果行為可以被精簡地描述，我們可以將可以期望其他權限程序將有精簡地描述正常行為。太多的變化將使得偵測異常困難，在最糟的情形，所有可能長度 k 的可能順序將可能判定為正常行為，沒有異常行為可以被偵測到。

長度 k 的有多少個可能順序呢？如果我們有 Sigma

的系統呼叫，其長度 |Sigma|，則有 $|Sigma|^k$ 個可能的順序。選擇字母長度(Alphabet Size)，在不知道 Sendmail 精確地使用多少個系統呼叫，仍可能是有問題的。在 SunOS 4.1.x 作業系統，我們假設 Sendmail 使用不超過 53 個系統呼叫，以長度 k=10，有 53^{10} 或大約 10^{17} 種順序可能。而 Sendmail 正常資料庫只包含大約所有可能特徵數的 10^{-13} 百分比。當然這些正常資料庫並非完全正確，因為可能順序的數量，實際上可以使用結構的碼被限制。

2.3.4.2 偵測異常行為(Detecting Anomalous Behavior)

定義正常行為下何種行為是入侵行為？本篇輸出未符合(Mismatch)的數量、未符合的百分比、與正規化的異常訊號 $\hat{S}_A$。因 $\hat{S}_A$ 與追蹤的長度二者並沒有相依性，在 $\hat{S}_A$ 值在上下文偵測的門檻是有意義的，門檻是與 False Positive 可接受程度有相依關係。為架構正常，我們有假造的資料且沒

有 False Positive(Zero False Positives)。任何 $\hat{S}_A > 0$ 表示異常。

2.3.5 不同程序間的差異

在其他的 UNIX 程式執行比較 Sendmail。假如我們無法區別 Sendmail 與其他程式，則將無法區別差異性更小的程式。本篇已做任何順序長度變化的比較，當順序長度分長低(Low)，k =1，將有很少的 Mismatch，其範圍介於 0 到 7%。當順序長度達到 k=30 則有 100% Mismatch。比較結果 k=10 表示如下：

表 2.3 程序狀態表

Process	Number Mismatches	% Mismatches	$\widehat{S_A}$
ls	42	75	0.6
ls -l	134	91	1.0
ls -a	44	76	0.6
ps	539	97	0.6
ps -ux	1123	99	0.6
finger	67	83	0.6
ping	41	57	0.6
ftp	271	90	0.7
pine	430	77	1.0

異常順序的特徵數至少 57，由 Sendmail Sequence 至

少有一個異常順序與正常的 Sendmail Sequence 相當不同，$\hat{S}_A$ 值至少 0.6，表示大部分異常的順序不同於正常順序超過一半的位置。

2.3.6 偵測入侵

本篇的實驗結果為入侵偵測以揭露 Sendmail, Lpr, 與 Wu.ftpd 三種程序的弱點。一些入侵有成功，其他未成功因為軟體有更新(Update)與修正(Patch)。本篇將可偵測大部分企圖入侵的行為，甚至失敗時也可發現。偵測入侵失敗對於入侵者企圖攻擊系統的重要訊號是有用的，第三個行為分類，本篇將可偵測錯誤狀態的發生，例如 Sendmail Forwarding Loops。雖然這些錯誤狀態是合法行為，他們視為不正常因為它們存在一些問題。

比較系統呼叫追蹤其分類有三種 Successful Exploits，Unsuccessful Exploits, 與 Error Condition，對於相關程式的正常資料庫與記錄 Mismatch 的數量，與追

蹤上 Mismatch 的百分比及$\hat{S}_A$值。

表 2.4 Sendmail 偵測成功情形表

Anomaly	Number Mismatches	% Mismatches	S_A
syslogd	248 - 529	17 - 30	0.7
sunsendmailcp	92	25	0.6
decode	7 - 22	1 - 2	0.2 - 0.5
lprcp	242	9	0.5
ftpd	496	38	0.7

Table 4 展現成功入侵的結果。本篇收集成功的入侵，其中有 Sendmail, Lpr[LGM94] 與 Ftpd[CERT93]。三種 Sendmail 入侵是：Sunsendmailcp[LGM94], Syslogd[LGM95][KFL94]與解碼別名入侵(Decode Alias Intrusion)。大部分成功的入侵可被偵測，$\hat{S}_A$值在 0.5 到 0.7。在解碼入侵有較低的範圍，只有 7 個 Mismatches，$\hat{S}_A$值在 0.2。這些結果有建議大約的偵測門檻值，以提供線上系統入侵偵測使用。

偵測未成功的入侵與錯誤情形如下表：

表 2.5 Sendmail 偵測失敗情形表

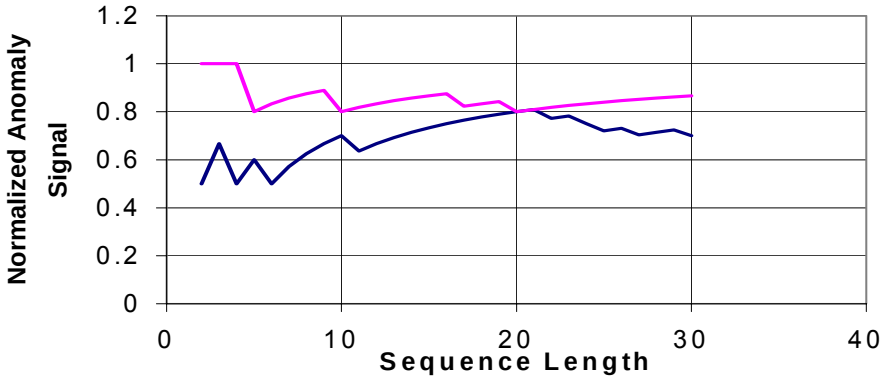
Anomaly	Number Mismatches	% Mismatches	$\widehat{S_A}$
sm565a	54	22	0.6
sm5x	472	33	0.6
forward loop	21 - 108	10 - 18	0.4 - 0.6

未成功的入侵是以攻擊的程序(Attack Scripts) 為基

礎稱為 sm565a 與 sm5x。 SunOS 4.1.4 已經 Patch 前面的入侵。這些未成功的入侵可被視為成功的偵測到入侵。錯誤的情形。錯誤情形也可被偵測在$\hat{S}_A$相似的範圍內。

綜合以上，本篇可偵測到所有異常行為，包含成功的入侵、失敗的入侵企圖與不正常的錯誤情形。

分析長度 k = 2 到 k = 30 的順序，最小的長度是 2，因為 k = 1 將使得$\hat{S}_A$= 0 或$\hat{S}_A$= 1 沒有充分的資訊。



解碼入侵於 k < 6 無法偵測到。增加順序長度(Sequence Length)結果在減少異常訊號產生。系統呼叫組成的異常短塊(Short clump)被大的 gap 分開，順序長度增加、較長的順序較類似於正常順序。例如：正常的順序 open, read, mmap, mmap, open, read，入侵破壞前 3 個位置，close, close,

close, mmap, open, read。

2.4 即時型專家入侵偵測模型

1987 年 Dorthy E. Denning 提出一個即時(Real-Time)入侵偵測專家系統模型能偵測到闖入(Break-in)、滲透(Penetrations)與其他形式電腦誤用。此模型包含表示行為主題(Subject)的概略檔(Profile)及其物件的矩陣(Metric)、統計及規則(Rule)的相關知識。因模型與任何的系統、應用環境、系統弱點、或入侵型態無關，故此架構為一般類型的入侵偵測專家系統(Intrusion-Detection Expert System；簡稱 IDES)[Den87]。

本模型發展的動機有 4 個因素，(1)大部分系統均存在安全弱點，使他們易受入侵、滲透、與其他形式的誤用所影響，發現與修正所有這些缺點，於技術上及成本上考量是不可行的。(2)系統存在已知弱點不易被系統置換，因為系統有吸引力的特點是缺少安全系統，基於成本的考量無法置換該系統。(3)正發展的系統絕對安全是困難的。(4)大部分的安全系統對於內部人

員誤用是有弱點。

本篇基本假設是系統弱點包含不正常使用(Abnormal Use)，因此違反安全將可以由系統使用上的異常特徵被偵測到，說明如下：

(1) 企圖闖入(Attempted Break-in)：

某一個人企圖闖入系統可能產生異常高的密碼失敗，可能針對某個帳號或系統整個帳號。

(2) 偽裝或成功闖入(Marsquerading or Successful Break-in)：

某個人使用未經許可的帳號與密碼登入到系統，其登入有不同的登入時間段、位置或連接時間。侵入者行為與合法者不同，如入侵者可能花費大部分時間瀏覽目錄與執行系統的狀態指令，而合法使用者則集中在編修、編譯與連結程式。

(3) 由合法使用者侵入：

一個使用者企圖滲入安全機制，在作業系統上將執行不同的程式或驅動更多的保護機制，以企圖存取未被授權檔案與程式。

(4) 由合法使用者洩漏：

使用者嘗試洩漏敏感的文件，這些行為可能記錄到系統。

(5) 由合法使用者推論：

使用者企圖獲得未授權的資料，藉由資料庫總計與推論將可存取比一般使用者更多紀錄。

(6) 木馬：

木馬的行為植入或替代程式在使用 CPU 時間與 I/O 活動上，將不同於合法程式。

(7) 病毒：

病毒的植入系統將發生在執行檔案的重覆讀寫頻率將增加、執行檔案將使用儲存空間或特殊的程式被執行，以作為病毒的分散的踏板。

(8) 阻斷式服務(Denial-of-Service)：

入侵者可能獨佔資源(例如：網路)，即有很高的不正常活動佔用資源。

即時型專家入侵偵測模型有六個主要元件：

(1) Subjects:在目標系統正常的使用者活動的發起。

(2) Objects:資源管理如系統檔案、指令、設備等。

(3) 稽核紀錄(Audit Records)：由 Subject 產生去執行活動或由 Subject 企圖在活動接收者(Object-user)登入、下指令執行、檔案存取等。

(4) 概略檔(Profiles):為 Subject 的行為特徵，到 Objects 的行為使用統計矩陣與觀察活動的模型(Models of Observed Activity)。

(5) 異常紀錄(Anomaly Records)：當不正常行為產生可被偵測。

(6) 活動規則(Activity Rules)：當某些情形滿足時，將更新概略檔，以偵測不正常行為，若有異常則產生報表。

本模型可被視為規則型特徵比對系統(Rule-Based Pattern Matching System)。當有稽核紀錄被產生時，再與概述相比對找出相關的資訊，最後決定是否修正概述規則，或輸出偵測出之異常的行為。

2.4.1 目標系統與活動接收者

目標系統(Subjects)是在目標系統活動的起始者。Subject i 是終端使用者，但也可能處理使用者或群體使用者的行為活動，或可能系統它自己。所有活動將 Subject 透過指令起始(Commands Initiated)。Subject 將可分群為不同的類別例如使用者群，以控制在系統中控制群取 Objects 的目的。使用者群可以重疊。

Objects 是活動的接收者，其包含的元件(Entity)如 Files、Programs、Messages、Records、Terminals、Printers 與 User 或 Program-Created Structures。當 Subject 可以是活動的接受者(例如電子郵件)，則這些 Subjects 也可以被考慮為在模型上的物件。

2.4.2 稽核紀錄

稽核紀錄使用 6 維度表示活動，這些活動由 Subject 執行在物件上(Objects)：

<Subject, Action, Object, Exception-Condition, Resource-Usage, Time-stamp>

其中

(1) Action:由 subject 的操作與執行，或作用的物件例如：系統登入、系統登出、讀取、執行。

(2) Exception-Condition:將有實際的例外情形由系統產生，不只是明顯的例外情形回傳給 subject。

(3) Resource-Usage:列出若干的元素，每一個元素給予使用一些資源的量，例如：多少行或頁被印出來，多少紀錄被讀或寫，CPU 時間或 I/O 單位被使用，會談(session)使用時間。

(4) Time-Stamp:當活動被佔用時，唯一 時間/日期時戳被識別。

我們假設每一個欄位是自我識別，是 Implicity 或 Explicitly，例如：Action 欄位隱含著期望的物件欄位或其他物件欄位它自己指定的型態。

大部分系統的操作包含多個物件，例如：編譯包含編譯程式，來源程式檔案，目的程式檔案，和可能的暫存檔，與參考的額外來源檔案。

下列為稽核紀錄產生如下的命令:

COPY GAME.EXE TO <Library>GAME.EXE

由使用者 Smith 欲拷貝一個執行遊戲檔到<Library> 目錄，

結果拷貝失敗，因為 Smith 沒有寫的權限到<Library>：

(Smith, execute, <Library>COPY.EXE, 0, CPU = 00002, 11058521678)
(Smith, read, <Smith>GAME.EXE, 0, RECORDS = O, 11058521679)
(Smith, write,<Library>GAME.EXE,write-viol,RECORDS=O,11058521680)

目標系統是負責稽核與傳送稽核紀錄到入侵偵測系統做分析。

稽核紀錄存在的另一個問題，包含一些或描述資訊以識別其價值。每一次記錄型態有自己的結構，與每一個紀錄型態有正確的型態，必須是中斷的值。一個制式的格式有自我辨識的資料，將較喜好入侵偵測軟體是系統獨立。他將可達到修改軟體產稱稽核紀錄在目標系統或藉由寫的過濾將紀錄轉換標準格式。

2.4.3 活動紀錄描寫檔

活動概述將行為特徵化，給予 Subject 與相對應的 Object，因此將特徵或正常活動的行為分開為 Subject 與 Object。觀察行為被特徵化在統計矩陣與模型。一個矩陣是亂

數 x 表示一段時間累積量化的衡量。期間可以是固定的一段時間(Minite, Hour, Week 等），或兩個紀錄相關事件(Audit-Related Event)的期間(亦即系統登入與系統登出期間，程式起始與終結的期間，檔案開啟與檔案關閉的期間等)。觀察樣本點 x，由稽核紀錄 x 所獲得，與統計模型決定是否新的觀察值是正常。統計模型在 x 分配上沒有假設，所有知識均由相關觀察之 x 得知。在描述架構、產生與概述的應用，將先由統計矩陣與模型說明如下。

矩陣，定義三種型態的矩陣：

(1) 事件計數器(Event Counter)： x 是稽核紀錄的數字，滿足一些特性在一段期間(每一個稽核紀錄相對應一個事件)。例如：在一個小時，在執行系統登入時指令的執行次數，與在數秒間有多少次的密碼登錄失敗數。

(2) 區間計時器(Interval Timer): x 是時間的長度，即不同的時戳有個別的稽核記錄，連續登錄到帳號其時間的長度。

(3) 資源衡量(Resource Measure)：x 是在一段期間資源的量由

一些動作所消耗，指定在稽核紀錄的資源使用(Resource-Usage)的欄位。例如：一天有多少使用者印出頁的總量與在執行程式時消耗 CPU 的時間。資源衡量在入侵偵測被製作一個事件計數器或目標系統。例如：有數個頁被印出在系統登入期間被製作在目標系統，當作事件計數器，以計算數個系統登入與系統登出間的列印事件，CPU 時間消耗由程式當做區間計時器，在程式的起始與終結記錄。因此，事件計數器與區間計時器衡量事件在稽核紀錄層級(Audit-record Level)，資源衡量需要資料由事件在目標系統，發生在稽核紀錄下面的層級。稽核紀錄的資源使用欄位，提供資料簡化的意義，所以比較少的事件被記錄在明顯的稽核紀錄上。

2.4.4 統計模型

給予亂數 x 的矩陣，與觀察值 $x_1, ..., x_n$ ，其主要 x 統計模型的目的，是決定是否有新的觀察值 x_n+1 ，與前面的觀察值是否正常。下面模型將包含在 IDES：

(1)操作模型(Operation Model)：此模型有如下的操作基本假設：經由計算新的觀察值 x 與固定的限制，不正常可以被決定。雖然前面樣本點對於 x 沒有使用，相同型態的變數與前面的觀察值其限制被決定。操作的模型大部分應用矩陣其經驗被展現在某些特定的值，大部分與入侵有關。事件計數器，針對在很短的期間內密碼失敗的次數，若超過 10 次將被視為入侵。

(2)期望值與標準差模型(Mean and Standard Deviation Model)：此模型基本的假設，所有已知 $x_1, \ldots x_n$ 其期望值與標準差由兩個時刻被決定：

Sum = $x_1 + \ldots + x_n$

Sumsquares = $x_1^2 + \cdots + x_n^2$

Mean = sum/n

Stdev = sqrt(sumsquares / (n+1)- mean^2)

一個新的觀察值 x_{n+1} 被定義為不正常，如果是落在信心度外，標準差由期望值與一些參數 d

Mean + d * stdev

藉由 Chebyshev's 不等式，機率的值大部分落在 $1/d^2$， d = 4，幾乎在 0.0625，0 將會被包含，所以是不偏資料。

此模型應用事件計數器、區間計時器、資源衡量建構。操作模型的兩個優點，第一、不需要有關正常活動的事先知識以設定限制，它知道由它的觀察值來替換正常活動，在增加知識時，自動反應信心區間。第二、因為信心區間依靠觀察資料，此觀察資料是考慮一個使用者正常行為推理另一個使用者。

在平均數與標準差模型輕微的變動是可以權重計算，使用大的權重計算最近的值。

多變數模型(Multivariate Model)：除了以兩個或更多的矩陣相關外，此模型類似於平均數與標準差模型。在實驗資料上這些模型是比差異量有益的，藉由結合相關的衡量，比各別好。CPU 時間與 I/O 單位使用，由程式、登入頻率與會談結束時間(Session Elapsed Time)。

馬可程序模型(Markov Process Model)：此模型應用事件計數器，是每一個唯一的型態事件為狀態變數，使用狀態轉換

矩陣到特性化,，狀態間的轉換頻率(比個別 State-i 的頻率好，亦即稽核紀錄是分開的)。新的觀察被定義在異常如果它的機率被前面的狀態決定與轉換矩陣太低。這些模型可能是有用的，將某些命令間轉換，其順序是非常重要的。

時間序列模型(Time Series Model)：此模型與使用區間計時器與一起的事件計數器或資源衡量，考量其順序與觀察值 $x_1, ..., x_n$ 區段時間的值。新的觀察值是不正常的，如果在時間上發生的機率是太低。時間序列有行為衡量的優點與逐步的偵測，排除在行為上特徵的移動(Significant Shift)。其缺點是成本比平均值與標準差高。

2.4.5 概略檔結構(Profile Structure)

活動概略檔包含資訊有識別統計模型與亂數的矩陣，藉由變數其稽核事件集合的衡量。概略檔的結構包含 10 個部分，前面 7 個是與特定的 Subject 與 Objects 衡量獨立的：

<Variable-Name, Action-Pattern, Exception-Pattern, Resource-Usage-Pattern, Period, Variable-Type, Threshold,

Subject-Pattern, Object-Pattern, Value>

1.Subject 與 Object 獨立的元件:

(1) 變數名稱(Variable-Name)。

(2) 活動特徵(Action-Pattern)：特徵其匹配值為(Match)零或更多活動在稽核紀錄，例如“login”, “read,”“execute”。

(3) 例外特徵(Exception-Pattern)：特徵與稽核紀錄的例外情形欄位相符。

(4) 資源使用特徵(Resource-Usage Pattern)：特徵與稽核紀錄的資源使用(Resource-Usage)欄位相符。

(5) 區間(Period)：衡量的時間區段，例如：day, hour, minute 。如果沒有固定的時間區間這個部分將是 Null。

(6) 變數型態(Variable-Type)：資料型態的簡寫名稱，矩陣定義簡單的型態與統計模型，例如事件計數器使用期望值與標準差模型。

(7) 門檻值(Threshold)：參數定義限制，使用統計測試決定異常。此欄位與它的解譯是由統計模型決定變數型態

(Variable-Type)。對於操作模型，它是一個觀察值的上限(也有可能是下限)。

2.Subject 與 Object 相依的元件：

(1)Subject-Pattern：特徵值與在稽核紀錄的 Subject 欄位相符。

(2)Object-Pattern：特徵值與稽核紀錄的 Object 欄位相符。

(3)Value：目前的觀察值與參數，使用統計模型值以表達前面值的分配。對於期望值與標準差模型，這些參數經過 Count，Sum, 與 Sum-of-Square。這些操作模型不需要參數。

概略檔唯一識別是由變數名稱、Subject-Pattern 與 Object-Pattern 決定。所有概略檔的元件除值外，其餘均是變動的。

雖然特徵的模型沒有指定正確的格式，我們可由下列識別：SNOBOL-like 結構如下：

‘sting’ String of characters

*	Wild card matching any string
#	Match any number sting
IN(list)	Match any string in list
P -> name	The string matched by p is associated with name
Pi p2	Match pattern p1 followed by p2
Pi | p2	Match pattern p1 or p2
Pi, p2	Match pattern p1 and p2
Not p	Match anything but pattern p.

特徵值實例：

‘Smith’
* -> User ---------- match any string and assign to User
‘<Library>*’ ----------matchfiles in <Library>directory
IN(Special-Files)-------match files in Special-Files
‘CPU=’ # -> Amount –match string ‘CPU=’ followed by integer, assign integer to Amount

下列概略檔的例子，以衡量輸出的量給終端使用者 Smith，並以會談(Session)為基礎。定義資源衡量係由活動資源變動表示，使用期望值與標準差模型。

Variable-Name:	SessionOutput
Action-Pattern:	“logout”
Exception-Pattern:	0
Resource-Usage-Pattern:	“SessionOutput=” # -> Amount

期間(Period):

Variable-Type:	ResourceByActivity
Threshold:	4
Subject-Pattern:	‘Smith’
Object-Pattern:	*

當入侵偵測系統接收到與稽核紀錄相符，一個變數的特徵值

其更新變數的分配檢查是否異常。

2.4.6 概略檔的類別化

概略檔可以被定義在一對 subject-object，即 Subject 與 Object 相符於指定的名稱，例如：Subject “Smith’ 與 Object “Foo’ 。或 Subjects 與物件的總合(Aggregates)，即 Subject 與 Object 特徵值相符的集合名稱，例如檔案活動概略檔可以被個別使用者與檔案對產生，使用者群與個別檔案類別，描述如下：

(1) Subject-Object: 活動被執行由單一 Subject 在單一 Object 上，例如 user Smith-file Foo。

(2) Subject-Object Class: 活動由單一的 Subject 綜合所有在類別的物件。物件類別可能表示如同特徵值比對，在 Object 欄位的子欄位，它指定物件型態如同直接特徵值比對在物件名稱(eg, “*.EXE’ 指對所有執行檔)。或特徵值比對測試是否物件在一些名單上，例如：“IN(hit-list)”。

(3) Subject Class-Object: 活動執行在單一物件總合在所有

Subject 的類別上。例如：privileged user-directorty file <Library>, nonprivileged users-directory file <Library>。

(4) Subject Class-Object Class: 在類別上所有 Subject 之活動的總合，與在 class-privileged users-system files 的物件上，non-privileged users-system files。

(5) Subject: 活動執行由單一 Subject 總計所有物件。例如：使用者會談活動。

(6) Object: 活動執行在單一的 Object 總計所有 Subject。例如：密碼檔活動。

(7) Subject Class:活動的總計 在所有類別上的 Subject。例如：具權限使用者的活動，不具權限使用者的活動。

(8) Object Class: 活動的總計在所有類別上的 Object。例如：可執行檔案的活動。

(9) System：活動的總計在所有 Subject 與 Object。

由概略檔表示亂數對於一個類別可以被總計之活動有兩種方式：

(1) Class-as-a-whole activity：所有 Subject 的集合或物件在類別當作單一個體，與每一個亂數的觀察，來表示個體活動的

總計。例如：所有使用者的類別概略檔，表示每天登入到系統的平均次數，所有使用者將為單一的個體。

(2) Aggregate individual activity：Subject 或 Object 在類別被當作是唯一的個體，且每一亂數的每一觀察值，其活動代表一些類別的數字。所有使用者特性的類別概略檔，表示每一個使用者每天平均登入系統的次數。

因此 class-as-a-whole 活動可以由事件計數器(Event Counter)、區間計時器(Interval Timer)、資源衡量(Resourve Measure) 被定義。個別使用者的類別需要個別分開的矩陣紀錄活動的總計。因此要定義低階的概略檔，作為個人成員的類別。例如，平均每天登入系統的頻率，定義為在個別使用者登入系統概略檔的整日平均總頻率。衡量 class-as-a-whole 將可以被定義在低階的概略檔。

兩個總合的活動可使用於不同的目的入侵偵測系統。

(1) Class-as-a-whole activity：可依據類別揭露一些行為的特徵值是否正常。可執行程式檔案類別的變動頻率，例如：偵測病毒植入系統，由可執行的檔案被寫入以傳播病毒。另外，遠

端登入系統的頻率類別，對於偵測企圖闖入者是有用的。

(2) Aggregate individual activity：個別活動的總計用以表達一個使用者(或物件)是與其它使用者一致。此對於入侵偵測由新的使用者有變動的行為是有用。

2.4.7 概略檔的樣板(Profile Templates)

使用者帳號與物件可以動態的產生，機制需要產生活動概略檔，對於新 Subjects 與 Objects。有三種方法：

(2) 人工產生：由安全管理人員產生所有概略檔。

(3) 自動產生：對於新的使用者或物件產生所有概略檔。

(4) 第一次使用：當新的或舊的 Subject 第一次使用 Object 時，自動產生概略檔。

在第一方法中，部分的安全工作人員，有明顯的缺點需要手動干預。第二方法中，雖克服第一種方法的缺點，但仍存在如下的缺點，首先不能自動處理很多存在 Subject 與 Object 的起始情形。第二需要 subject-object 概略檔產生，用來監測使用的情形，甚至 Subject 在特別的 Object

不能使用。這將造成數量比需要更多的概略檔，例如：檔案存取是監控個別使用者層級與檔案,若考慮系統有 1000 使用者,每一個使用者平均有 200 個檔案,給予 200,000 個檔案，則有可能有 200,000,000 可能的 user-file pairs。如果每一個使用者最大 300 的檔案，只有 300,000 概略檔需要。

IDES 模型採用第三種方式,克服其他方法的缺點藉由樣版(Templates)產生概述檔。一個概述檔樣版有一些架構,例如概略檔除 Subject 與 Object 特徵值，定義在稽核紀錄上有兩個 Matching Pattern 與 Replacement Pattern，欄位的格式如下：

Matching-pattern <- replacement-pattern

在特徵值比對期間，其特徵值定義是動態的。概略檔樣版(Template Profile)的元件值包含變數開始的值，如同被指定的型態。

當有新的稽核紀錄被接受，處理比對紀錄、活動概略檔與概略檔樣版、存在的概略檔、新的概略檔由概略檔樣版產生。此 Subject 與 Object 特徵值包含在比對期間重置的特

徵(Replacement Pattern)，所有的欄位被正確的由樣版拷貝。如果新的樣版有相同的特徵，如同已存在的活動概略檔，它可以被宣告，或加入活動概略檔的集合。程序(Process)回傳活動概略檔與稽核紀錄相比對。

分開的比對與置換特徵值需要樣版可以廣泛的比對整個 Subject 與 Object。例如：下列特徵：

Subject-Pattern: * -> user <-user
Object-Pattern: IN(Special-Files) --> file <-file

Subject 特徵將比對任何使用者名稱，且產生置換的特。類似地 Object 特徵將比對任何檔案，此檔案列在特別檔案(Special-Files)與產生置換特徵。假設特別檔案包含檔案名稱、密碼與帳號。下列稽核紀錄的順序與這些比對的概略檔樣版與置換特徵值將產生：

Audit Records		**Generated Profiles**	
Subject	**Object**	**Subject-Pattern**	**Object-Pattern**
‘Smith’	**‘Password’**	**‘Smith’**	**‘Password’**
‘Jones’	**‘Accounts’**	**‘Jones’**	**‘Accounts’**
‘Smith’	**‘Foo’**	**no match, so no profile**	

Subject 與 Object 特徵值在樣版與下列特徵值獨立：

Subject-Pattern: * -> user <- user

Object-Pattern: ‘<’ user ‘>*’ <- ‘<’user’>*’

Object 特徵值將比對任何在使用者目錄的檔案與產生概略檔供使用者目錄使用(如果尚未存在)。下列說明稽核紀錄的順序，與暫存檔產生以下特徵值的概略檔：

Audit Records		**Generated Profiles**	
Subject	**Object**	**Subject-Pattern**	**Object-Pattern**
‘Smith’	**‘<Smith>Game’**	**‘Smith’**	**‘<Smith>*’**
‘Smith’	**‘<Smith>Let’**	**no new profiles generated**	
‘Jones’	**‘<Jones>Foo’**	**‘Jones’**	**‘<Jones>*’**
‘Jones’	**‘<Jones>Foo’**	**no new profiles generated**	

新的使用者(與 Object)給目標系統，其潛在有兩個問題。第一有缺乏概略檔資訊如同系統沒有資訊，因此產生大量的異常紀錄。此問題將可於新使用者情形時，忽略異常來處理，但入侵者是新的使用者則無法正確偵測到。

錯誤警告(False Alarm)可以由選擇適當的統計模型來控制活動造成的警告，與選擇適當的概略檔。隨者所蒐集使用者資料的期望值與標準差模型，信心區間開始很大，所以資料有更多變動的容忍率。隨著觀察值的增加區間會減少。藉由個別使用概略檔將減少錯誤警告，但仍未保護系統對抗新的使用者(或使用頻率較低的使用者)，這些行為是不可靠的，或使用者建立不正常行為。處理這些問題可以比較目前的活動紀錄，將個別概略檔總計或使用所有概略檔集合或所有使用者分群。

雖然操作模型沒有自動採用個別使用者，因其是固定門檻決定是否異常行為，藉由新使用者寬大的限制，或藉由經驗調整範圍，以避免此項問題。

以下為描述候選概略檔以衡量系統登入與會談活動、命令與程式、與檔案存取的使用。本篇建議採用矩陣與統計模型衡量活動。更多的概略檔規格在 IDES[DN85]有詳述。

(5) 系統登入與會談活動: 系統登入與會談活動在稽核紀錄，表示 Subject 在何處; 使用者 i, 物件是使用者系統登入位置(終端電腦)、工作站, 網路, 遠端主機, 通訊埠等，這些動

作是系統登入或系統登出。位置(Location)將被分成群到類別，藉由連接型態的特性：hard-wired, dial-up, network 等，或位置型態：dumb terminal, intelligent workstation, netowork host 等。以下列出可能的概述檔：

(1) 系統登入頻率(Login Frequency): 事件計數器以衡量每日系統登入頻率與時間，使用期望值與標準差模型。因為使用者系統登入行為有各種變化的考量，在每周工作天，系統登入發生將會以一周的一天事件計數器參數的陣列表示(指定一天或平日與週末相比較)，一天的時間(小時)，另一個可能的分類：Weekday，Evening,Weeken,Night 之概略檔，對於登入的頻率將特別有用，對於偽造者在非上班時間使用未經授權的帳號以及合法使用者帳號在不可期望的時間使用。登入概略檔將被定義各別的使用者(與使用群)，Locations-either 類別，係將所有位置聚集或將位置型別進行總計。

(2) 位置頻率(Location Frequency): 事件計數器於衡量在

某個位置上的登入頻率使用平均數與標準差模型。一周的一日與一天的某個時間衡量將會失敗，因為使用者在正常工作時間或其他非工作時間，在一個位置登入，所以無法衡量比較。因為指定的物件的相關變化，將被定義個別位置與位置的型別，其可以使用偵測偽造者，例如某人由某個位置登入系統，而此合法使用者從未使用，或企圖以合法使用者滲透。又如某個人正常的工作由非權限的終端機登入到較高的權限終端。

(3) 最後系統登入時間(Last Login): 區間計數器衡量時間自最後一次系統登入使用的操作模型。概略檔的型態將可以被定義為個別使用者，而非位置類別，因為正確的位置似乎與終止時間少有相關。對於偵測闖入者使用“dead account”似乎很有用。

(4) 會談結束時間(SessionElapsedTime):每一個會過去資源的利用使用平均數與標準差模型。概略檔的型態被定義在個別使用者或群體，並非物件類別。變化情形將被表示為偽造者。

(5) 會談輸出(SessionOutput):用於資源衡量，每一個會談輸出到終端機的量，使用平均數與標準差模型，其輸出將可以一日為基礎。定義概略檔型態，以個別位置或所屬類別，進行偵測大量的資料傳送到遠端位置，其可以識別敏感性資料洩漏。

(6) SessionCPU, SessionIO, SessionPage：用於資源衡量，累積每日為基礎(或單位為基礎)，使用平均數與標準差模型。這些概略檔將可使用於偵測偽造。

(7) 密碼錯誤(PasswordFails):事件計數器紀錄登入系統密碼失敗。概略檔的型態，係用以偵測企圖闖入情形，藉由被定義個別使用者與所有使用者的情形。一個攻擊包含許多密碼嘗試(Trail Password)登入特殊的帳號，因此將出現一個不正常的高失敗率次數。對於個別帳號的概略檔，攻擊包含個別密碼嘗試在很多帳號，將不正常高失敗次數的密碼，紀錄於使用者的概略檔。密碼失敗將被記錄在相當短的時間區間裡，因為闖入者通常企圖有一個突然的活動。

(8) 位置失敗(LocationFails):事件計數器衡量失敗，係由特定的終端登入系統之操作模式。這種型態的概略檔可能定義在個別使用者，並非位置的總計。它將可被使用偵測企圖闖入或企圖登入至有權限的終端機。

(6) 命令或程式執行(Command or Program Execution): 命令或程式執行執行活動，是以使用者為 Subject 表示在稽核紀錄，Object 是程式的名稱。程式將會被分類與總計，分為是否有權限(即由具權限使用者執行，或具權限模式)，或無權限。

(7) 執行頻率(ExecutionFrequency): 事件計數器以衡量程式在一段期間被執行的次數，其衡量以平均數與標準差模型。概略檔的型態可以被定義為個別使用者與程式或所屬類別。對於個別使用者與命令，於偵測偽造時是有用的。偽造者由合法者身分並使用不同的指令，或利用由合法使用者滲透成功，進入後擁有權限。個別程式的概略檔，非所有使用者可以被使用，以偵測木馬替換。若實驗資料庫在標準資料庫前被搜尋，因為執行原來程式的執行頻率有異，而將被系統拒絕。

(8) ProgramCPU, ProgramIO, etc:Resource 衡量，每次程式執行，使用平均數與標準差模型。這種型態將可被定義為個別使用者與程式語類別所屬。這些衡量的不正常值，使用到所有的使用者總計(aggregate)，可能被偵測為被植入木馬或病毒。

(9) 執行拒絕(ExecutionDenied):事件紀錄器對於企圖執行，而未被授權的程式的紀錄在一天內簽入的操作模式。定義這些概略檔型態，個別使用者可能使用由某些特殊的使用者滲透偵測。這種型態的概略檔可能也被定義在個別程式，這些程式有高敏感，與 1 的門檻值很接近。

(10) 程式資源用盡(ProgramResourceExhaustion): 事件計數器記錄一天內因為資源不足，而程式不正常終結的次數。這種型態的概述檔可能針對個別程式被定義或程式類別去偵測程式，連貫的放棄(Abort)。

(11) 檔案存取活動(File-Access Activity): 檔案存取活動表示在稽核紀錄，其 Subject 是一個使用者，一個檔案的物件名稱，其動作有”read”, “write”, “create,”

“delete,” 或”append”。檔案以其型態分類：text, executable program, directory,等，藉由它們是否為系統檔案或使用者檔案，或其他特性。因為程式是檔案，它可以監控它的活動執行與檔案存取活動。

下列衡量對於概略檔的候選人：

(1)讀取頻率(ReadFrequency), 寫入頻率(WriteFrequency), 建立頻率(CreateFrequency),刪除頻率(DeleteFrequency):事件計數器是衡量個別型態的存取數量(或一些區間)，使用平均數與標準差模型。讀與寫之存取頻率概略檔將可被定義，對於個別使用者與檔案類別。建立與刪除存取概略檔，總計檔案活動因為個別檔案被建立與刪除最多一次。不正常的讀取與寫入頻率，使用者可以識別為偽造或瀏覽。

(2)紀錄讀取(RecordsRead),紀錄寫入(RecordsWrite):資源衡量係對紀錄讀取或寫入紀錄，每一個存取可以每天為基礎，並使用平均數與標準差模型。這種型態概略檔可被定義為個別使用者與檔案或所屬類別。一個不正常

將可識別為企圖獲得敏感性資料，藉由推論與總計。

(3) 讀取失敗(ReadFails), 寫入失敗(WriteFails), 刪除失敗(DeleteFails), 建立失敗(CreateFails):事件計數器用以衡量每天違反存取的個數。概略檔型態可能被定義個別使用者與檔案或所屬類別。概略檔是個別使用者與所有檔案的類別，其可能持續存取未授權檔案。概略檔個別檔案與所有使用者的類別，可以對未授權存取的高敏感度檔案偵測(其門檻集合是 1)。

(4) 檔案資源用盡(FileResourceExhaustion):事件紀錄器衡量失敗的數量，起因於企圖超過(Overflow)可使用的數量。 概略檔的型態將被定義為個別使用者的所有檔案的總計。此種型態的概略檔亦可被定義為在所有檔案之個別使用總計。一個不正常將可被視為不合法使用者透過秘密管道，消耗所有可能的磁碟空間。

2.4.8 異常紀錄(Anomaly Records)

透過活動的規則，IDES 更新活動的概略檔與檢查異常行

為，當稽核紀錄被產生或終結的區間。如果異常行為被偵測，異常紀錄將產生三種元件說明如下：

<Event, Time-stamp, Profile>

其中

(1) Event:代表事件增加不正常，資料被發現異常並被紀錄，或“period”，表示資料累積在此段區間被發現異常。

(2) Time-stamp:在稽核紀錄的時戳或區間停止時間，因假設稽核紀錄有唯一時戳，它提供異常背後的意義。

(3) Profile:活動概略檔其不正常被偵測,其可包含完整的概略檔, IDES 可能包含 “key” 欄位，其識別在資料庫的概略檔。

2.4.9 活動規則(Activity Rules)

活動規則，當稽核紀錄或異常紀錄被產生，或時間區間結束時產生，其包含兩個部份:情形(Condition)與本體(Body)。情形引發規則。當我們使用“body” 優於“action”，為避免混亂其活動由 IDES 偵測。此種情形被指定一個特徵值比對在事件發生。其有 4 種型態規則：

(1) 稽核紀錄規則(Audit-record rule)：在比對新的稽核紀錄與活動概略檔、更新概略檔兩個檔時觸發，此規則會更新概略檔與檢查異常行為。

(2) 週期活動更新規則(Periodic-activity-update rule)：由區間的比對的結束，此區間活動的概略檔、更新概略檔與檢查異常行為。

(3) 異常紀錄規則(Anombly-record rules): 在異常紀錄產生時驅動，異常行為將立即警告管理人員。

(4) 週期異常分析規則(Periodic-anomaly-analysis rule):在區間結束時驅動，在現在期間產生異常摘要報告。

稽核紀錄規則係在新的稽核紀錄與活動概略檔的特徵值比對時觸發。更新概略檔係反應活動在變動行為檢查的報告。如果異常行為被偵測，可以產生異常紀錄。因為演算法對概略檔更新與檢查不正常，依變動的型態(統計矩陣與模型)，表示概略檔，但在概略檔其他的元件(例如：subject, object, action 等)，它可以被編碼在稽核處理的程序。所有稽核紀錄規則以下列規則表示：

AUDIT-RECORD RULE
Condition: new Audit.Record
Audit.Record matches Profile
Profile.Variable-Type = t
Body: AuditProcess(Audit-Record, Profile);
END

(1) 週期活動的更新規則(Periodic-Activity-Update Rules)

此種型態規則是統計衡量的型態參數，clock 表示於 p 長度完整區間時被驅動。區間的元件是概略檔 p，變數型態元件 t。規則更新係檢查異常行為，與產生異常紀錄。

PERIODIC-VARIABLE-UPDATE RULE
Condition: Colck mod p = 0
Profile.Period = p
Profile.Variable-Type = t
Body: PeriodProcesst(Clock, Profile);
END

(2) 異常紀錄規則(Anomaly-Record Rules)

當有新的異常紀錄與特徵值相符時，每一個異常紀錄規則將被觸發，即給予一個規則，其規則有事件元件與概略檔。因此，規則可以情形(condition)為其特別的變數，一個特別的 Subject 或 Object，在稽核行動時被發現異常等。對於這些概略檔的元件，其也是特徵值(例如：subject 與 object 元件)，特徵值給予一個異常規則必須被識別，對於比對時，

一個特徵值比對另一個，如果特徵值被識別。比對相符的紀錄將被立即注意，其規則產生形式如下：

ANOMALY-RECORD RULE
Condition: new Anomaly-Record
Anomaly-Record.Profile matches profile-pattern
Anomaly-Record.Event matches event-pattern
Body: PrintAlert('Suspect intrusion of type…', Anomaly-record);
END

不幸地，我們有很少的知識有關某類形態不正常的正確關係與入侵。在那些情況，我們有經驗，我們可以寫規則納入我們的知識。例如：密碼輸入失敗，安全管理人員可以立即注意到闖入，如果系統有多個密碼失敗則為不正常。其他異常立即識別包含不正常終止，因為最後係登入或不正常登入時間或地點。又如使用者曾登入系統在深夜或晚間，此將可表示偽造。

(3) 週期異常分析規則(Periodic-Anomaly-Analysis Rules)

規則的型態由一段區間結束後觸發，其可以被分析一些異常紀錄的集合，對於區間與產生摘要異常報告，其一般型態如下：

```
PERIODIC-ANOMALY-ANLAYSIS RULE
Condition:                          Clock mod p = 0
Body:                               Start = Clock – p;
                                    A = SELECT FROM
Anomaly-Records WHERE Anomaly-Record.Time-stamp>
                                    Start;
                                    Generate summary report
of A;
END
```

規則選擇所有異常紀錄屬於該期間，由所有異常紀錄集合。規則處理異常可能產生統計的總合表單。為使異常報告更容易，模型可以被加強包含異常概略檔。異常概略檔可以類似活動概略檔，除更新外將可以被觸發，由 IDES 產生的異常紀錄較優於由目標系統的稽核紀錄。

IDES 模型提供一個基本，可發展更強大的即時型入侵偵測，其可偵測廣範圍的入侵，例如企圖闖入、偽造、系統滲入、木馬、病毒、洩漏給予其他合法使用者濫用、與一些秘密管道。此模型允許入侵被偵測，不需要知道目標系統的弱點，允許入侵發生，不需要觀察特別活動暴露的弱點。

下面有一些問題：

(1) 此一方法可以偵測入侵嗎？它可以區別不正常由入侵的相關因素嗎？

(2) 此方法偵測大部分不是全部入侵，或是明顯有不可偵測到的比率？

(3) 可以是否可以在明顯災害發生前被我們偵測到？

(4) 哪一種矩陣、模型、與概略檔提供最好的分別能力？哪一個有成本影響？異常的型態與不同方法的入侵的關係？

(5) 系統如何以模型被設計與製作？

(6) 在目標系統中入侵偵測的影響? IDES 應該自動直接採取某種活動？

(7) 一個入侵偵測系統如何影響他所監督的使用者？它防礙入侵嗎？使用者採用它的 data 是比較好的保護嗎？它視一個步驟朝向“big brother”？最終它有可能被誤用嗎？

雖然我們相信，此方法可以被偵測大部分的入侵，它可能是個別離開偵測，透過逐漸修改行為，透過入侵微妙格式，使用低階的目標系統特徵，此特徵不能監控(因為它們產生太多資

料)。例如：因為實際監測個別頁框失效(Page Faults)、程式洩漏資料、秘密地控制頁框失效將被偵測，藉由頁框失效活動。

偵測低階活動間，這些活動存在系統的弱點，偵測相關的入侵。例如：考慮操作系統滲入，嘗試具有管理者權限的系統呼叫，有微量變化的參數呼叫，直到參數被發現，它允許使用者執行其程式在管理者模式。偵測實際上的滲透將可能，沒有監控所有最高權限系統呼叫，故其產生不適實際。一旦滲透成功，如果入侵者開始存取物件前，入侵將被偵測。因此重要的問題為是否我們可以偵測特殊低階活動。

2.5 適應性模糊類神經入侵偵測系統

入侵偵測系統架構一般使用在商業上與研究上的系統，有數個問題，以限制其設定(Configurability)、擴展性(Scalability)、效率(Efficiency)。在本篇有兩個機器學習推理系統(Machine-Learning Inference System)，使用設計一個入侵偵測系統。SNORT 被使用執行即時流量分析與 IP 網路封包登入，在系統訓練階段，其特徵資料庫被建構於使用網路協定

分析與模糊類神經學習方法(Neuro-Fuzzy Learning Method)。其使用 1998 DARPA 入侵偵測所公開的資料集 TCP 原始資料。

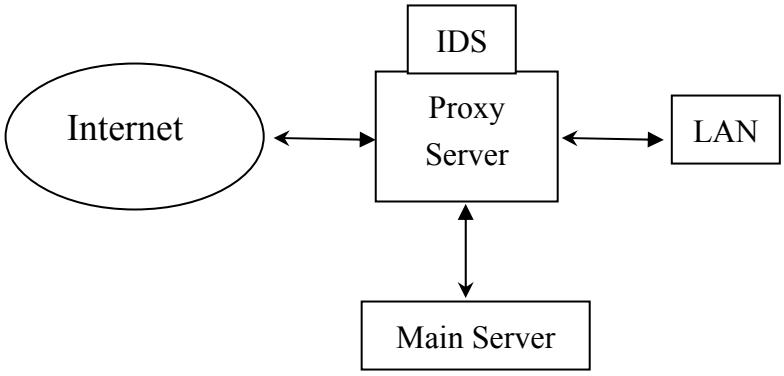


圖 2.9 適應性模糊類神經入侵偵測系統架構圖

2.5.1 系統架構

本篇將系統元件包含如下列：

(1)區域網路(LAN):一個可信賴乙太網路拓樸的網路。

(2)主機/伺服器(Host/Server): 位於內部網路。

(3)網際網路(Internet): 外部的閘道器(Gateway)。

(4)代理伺服器(Proxy Server): 代理伺服器是介於工作站與網際網路中間，以確保伺服器安全、管理控制與暫存服務(Caching Service)。

(3)入侵偵測系統(IDS): 比較入侵偵測系統模型在[MSA03] [MSA]，其所提出的入侵偵測系統可以連續學習新型態的攻擊，而不會刪除前面訓練的知識。以此為基礎，入侵偵測系統可以更新知識庫，使用 if-then 模糊規則。

2.5.2 代理伺服器

代理伺服器是一種服務，其檢查應用或服務封包之意圖，如果此服務可使用，則被允許通過，因此不可信任區沒有直接與信任區系統連接。

首先，代理伺服器是行為媒介物，幫助使用者在私密網路獲得在網際網路上的資訊。第二，代理伺服器可以快速的儲存，可向當地磁碟暫存區(Local Disk Cache)要求資訊，故可快速地傳送給多個使用者，不需要回到網際網路抓取。

代理伺服器執行在網路位址轉換(Network Address Translation)，對應所有網路內部 IP 位址到單一且安全的 IP 位址。

2.5.3 網路型入侵偵測系統

網路型入侵偵測系統(Network Based Intrusion Detection System)，其由 3 個子系統組成：輸入子系統、處理資系統、輸出子系統。輸入子系統包含，Snort IDS 如同封包的 Sniffer 與特徵值資料庫。處理子系統使用 Misuse 與 Anomaly 偵測技巧，並結合類神經使其可以調整。輸出子系統使用各種輸出機制。

藉由 Snort 我們可以存取資料，其資料是訓練階段的輸入。Snort 被使用於訓練階段的演算法，是一個 Libpcap-based Sniffer 與紀錄器(Logger)[LCP00]，也是跨平台、Lightweight 的入侵偵測工具，它可以部署以偵測小的 TCP/IP 網路，亦可偵測廣泛可疑的網路流量，如同公開的攻擊。另外也管理足夠的資料，提供資訊以便決定採取何種措施。偵測的引擎採用簡單的語言，以描述每一封包測試與行為。SNORT 主要特徵是檢查封包內容(Packet Payload)。

2.5.4 特徵資料庫與協定分析

於 Snort 中參考資料庫存在的特徵,此資料庫試著合併措施(Cures)與弱點。網路 IDS 特徵是搜集網路流量所萃取分析得到。如下的一些例子與一些方法可以識別特徵。

(1) 企圖連結保留之 IP 位址,由檢查非法 TCP 旗標的 IP 封包頭與封包之來源位址欄位。

(2) IDS 比較每一個電子郵件的標題(Subject),標題關聯到病毒郵件,或以特殊的名字尋找附件。

(3) 在 POP3 伺服器的阻斷式攻擊,藉由發動數百個相同的命令。此類攻擊有一個特徵值可追蹤多少命令被發送,當超過某一個門檻值時發出警告。

(4) 檔案存取攻擊在 FTP 伺服器,藉由發送檔案與目錄命令,而無須登入系統。狀態追蹤的特徵可以被發展,以監控 FTP 流量,如果某一個命令被發送將發出警告。

因為我們已識別四種潛在的特徵元素,我們有很多不同發展

封包頭基礎的特徵(Header-based Signature)，因為特徵可以包含任何一個或更多的特徵。在特徵資料庫中，所有 4 個可疑的特性，雖然這些提供更精準的資訊。有關來源活動，它可能比檢查封包頭的值更沒有效率。特徵之發現常於效率(Efficiency)與正確性上(Accracy)進行取捨。在許多情形，簡單的特徵將產生 False Positive，因為簡單的特徵更具一般性。但複雜的特徵將發生 False Negative 比簡單特徵更多[Lau02]。

雖然流量的特性改變，透過使用一般的特徵，仍然可以識別異常。本篇主要使用通訊協議的分析策略，可以尋找一般的特徵，並發展以人工智慧為基礎的自我學習 IDS。

“通訊協定分析”，表示 IDS 的監測器(Sensor),以了解各種通訊協定如何運作，與這些通訊協定接近的分析，以尋找可疑或不正常的活動[MHL94]。許多封包頭數值可以被使用產生網路 IDS 特徵。有些一般性的封包頭相關的特徵值[Cox94]，說明如下。

(1) IP 位址：保留位址(Reserved)、不可路由

(non-routable)、廣播位址(Broadcast Address)。

(2) 不應該被使用的通訊埠：著名已知的通訊埠與特殊的通訊協定及木馬有關。

(3) 非一般性的封包切割。

(4) 特別的 TCP 旗標的結合。

(5) 非一般正常可看到的 ICMP 型態。

藉由觀察流量的異常，比簡單地尋找某個特徵值更有效，通訊協定分析基礎的特徵值(Protocol Analysis-based Signature)對於攻擊者修改病毒碼情形，仍可輕易地發現。

2.5.5 機器學習的架構

各種非線性系統已經被提出，其具有取出需要的特徵值或回存特徵值，其結果可以計算在一個時期，或逐步更改動態的公式。本篇採用適應性機器學習演算法(Adaptive Machine Learning Algorithms)發展 IDS。類神經網路的突出特性是具有學習能力，其可以更新權重，用以加強連結，

其權重更新是依據新的訓練特徵萃取的資訊變更。

模糊推理系統 FIS(Evolving Fuzzy Inference System)可以利用人類專家、儲存在規則資料庫、資料庫的基本元件，與執行模糊推理整個輸出值。 If – then 規則的起源與相對應的隸屬函數(Membership Function)，主要考量在系統上的事前知識。然而，沒有系統可轉換人類專家的知識經驗到 FIS 的知識平台，因其需要適應性調整(Adaptability)或一些學習的演算法以產生輸出。

為大量的擴展，這兩個方法的缺點，似乎成為對比，因此可結合 FIS 與 ANN 模型[Man94]

設計模糊類神經網路 EFuNN (Evolving Fuzzy Neural Network)製作一個 Mamdani 型態 FIS[MA75]，在學習期間所有的神經元將被建立[Kas98]。神經元表示成隸屬函數 可以在學習過程中修改。每一個輸入變數在此代表由群體空間安排神經元(Neurons)表示這些變數的模糊量子化。不同於這些模糊函數，可以被隸屬於這些神經元(Triangular, Gaussian 等)。新的神經元可以被設計在這一個階層(對於

給予的輸入矩陣)，其相對應的變數值不屬於任何存在的隸屬函數，其程度大於隸屬門檻值。一個新的模糊輸入神經元，或輸入神經元，可以被建立在 EfuNN 的適應性階段學習的演算法其技術細節在[Kas98]。

2.5.6 系統製作

階段 1: 訓練演算法

對於任何機器學習基礎的演算法，需要一個好的訓練資料集以獲得較佳的解決方式，即可藉由大量的輸入集合以進行訓練(它可以是有攻擊或沒有攻擊的資料)，它可使用 Snort 獲得資料。隨著大量的攻擊資料庫，其資料庫是類神經與 FIS，經訓練階段累積後，以使所發展的知識庫於未來能識別攻擊。

階段 2: 執行階段

為達成上面的目標，本方法所建立的參考資料庫其特徵值包含 Snort 上之弱點，輸入參數由 TCP Dump 萃取出。藉由機器學習演算法學習與建立特徵資料庫，故可辨識是否為攻擊

或正常封包。

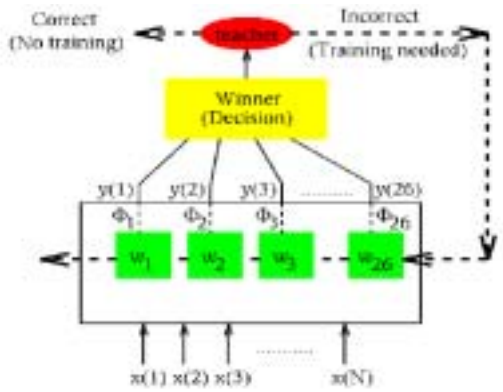


圖 2.10 多類別攻擊學習架構

2.5.7 類神經網路學習

子網路被設計為每一攻擊的類別。線性差異函數(Linear Discriminant Functions)對於子網路被定義為 $\Phi(x, w_i)$, i = 1, …, L。差異函數提供每一子網路(或類別)的分數。有一個程序(Procedure)被使用於選擇優勝分數(Winning Score)的子網路，輸出通常被標示為優勝的子網路。下列共同的訓練方法被使用[Lau91][Fau94]。如果網路未符合,則權重將由 Reinforcement 與 Anti-reinforcement 學習規則

進行更新。假設 $S = \{x^{(1)}, \ldots, x^{(M)}\}$ 是給予訓練特徵的集合，其中每一個元素 $Z^{(m)} \in R^N$ ，屬於 L 類別的一個 $\{ \Omega_l, l = 1, \ldots, L \}$, 差別函數 $\Phi (x, w_l) = w_i^T z$, for i = 1, …, L。假設第 m 個特徵值 $x^{(m)}$ 被表示為屬於類別Ω_i,則特徵值的成功類別(Winning Class)被定義為整數 j，對於所有 $l \neq j$，$w_j^T > w_l^T Z$ 。

1. 當 j = i ，則特徵值 $z^{(m)}$ 已經被正確分類，所以不需要更新。
2. 當 $j \neq i$, $z^{(m)}$ 仍未被正確分類，則下列的更新將被執行：

Reinforcement 學習: $w_i^{(m+1)} = w_i^{(m)} + z^{(m)}$

Anti-reinforcement 學習: $w_j^{(m+1)} = w_j^{(m)} - z^{(m)}$

其他權重剩下的未改變：

$w_l^{(m+1)} = w_l^{(m)}$, 對於所有 $l \neq i$ 與 $l \neq j$.

2.5.7.1 實驗設定與結果

為模擬這一個想法，本篇使用 1998 年 DARPA 入侵偵測程式資

料， 由 MIT Lincoln Labs [MIT]，大約有 5 百萬的紀錄被本篇使用，其中資料集包含有 24 種攻擊型態，其攻擊主要可分為四類：

(1) 阻斷服務 DOS (Denial of Service)：攻擊者使一些計算或記憶資源過於忙碌，以致無法服務合法的使用者。

(4) 使用者遠端登入(R2L)：攻擊者在遠端機器雖沒有帳號，但透過網路發現系統弱點，利用系統弱點以獲得該機器的合法登錄。

(5) 使用者取得管理者權限 U2R (User to Root)：攻擊者開始存取在系統上正常使用者的帳號，可以利用系統弱點，以獲得系統的 root 權限。

(6) 探測(Probing): 攻擊者掃瞄電腦網路整合資訊，以發現系統已知的弱點。攻擊者對於網路上的機器及服務有一個地圖，利用此資訊可以尋找其弱點。

原始的資料包含 744MB，其有 4,940,000 筆紀錄。資料集有 41 個屬性，對於每一個連接紀錄加上類別標籤，一些特性衍生出特徵，其對於區別正常連結與攻擊封包有用。另外，一些

特徵檢查只連接過去兩秒鐘，其連接到相同的目的地電腦，計算與通訊協定相關行為的統計、服務等，此被稱為相同主機的特徵。相同的主機與相同的服務特徵一起衡量，被稱為連接層上以時間為基礎的流量特徵紀錄。本篇起初的研究是降低變數的數量，使用所有 41 個變數，可以使用於大的 IDS 模型，並可以使用於線上偵測上。本篇產生決策樹以決定變數的重要。

對於節點 n，如果攻擊者出現如同主要分裂(Primary Splitter)，則他有使 x 增加重要性。如果代替攻擊者(Predictor)出現在第 n 個代理人，以代理人代替主要攻擊者，其貢獻變為 $p^n \times X$，其 p 是代理人增加其權重，其於任何地方介於 0 到 1。IDS 模型主要目的，是分類資料集到 4 類的一種或正常。本篇實驗中，資料集包含 11982 筆紀錄，其由資料集亂數產生[KDD99]。此資料集 有五種不同的類別，亂數產生的資料包含資料的數目，由每一類別成比例得到其的長度。此資料集分兩個訓練資料為 5092 紀錄與 6890 筆測試紀錄。所有 IDS 模型被訓練與測試相同的資料集，其實驗包含兩個步驟：網路訓練與評估效能。所有訓練資料介於 0-1。決策樹方法可減少變數到 13, 14, 15,

17 與 16，分別代表 正常、阻斷服務、U2L、U2R 與探測。

2.5.7.2 EFuNN(Evolving Fuzzy Neural Network)訓練

本篇使用 4 個隸屬函數，其偵測所有攻擊類別如下：

敏感性門檻值 Sthr =0.95 與錯誤門檻值 Errthr = 0.05。

在訓練階段，發展 89,115,123,134, 與 129 規則節點分別為正常、DOS、U2L、U2R 與探測。

2.5.7.3 類神經網路訓練(ANN Training)

本篇使用 80 隱藏神經元(Hidden Neurouns)與輸入神經元(Input Neurons)的數目，其輸入變數與一個輸出神經元其開始的權重，學習率(Learning Rate)與增加量(Momentum)是 0.3, 0.1 與 0.1，訓練終止在 4500 epochs 之後。

Table 1 比較執行率(不同攻擊型態的分類正確性) 在 EfuNN 與 ANN 的測試資料庫。當 EfuNN 在訓練 IDS 模型上花費較少的時間，ANN 花費較多的時間。除了 U2R 可以利用模糊推理系統使其偵測的正確性較高。當所有的變數均使用

時，執行效能被降級。

FIS-based IDS 很容易使用簡單 if-then 規則，這些規則可以由資料中自動地學習，使得 EFuNN(Evolving Fuzzy Neural Network)成為理想的適應性學習候選。我們可以加特徵與相對應的新弱點與加強資料庫與加強 IDS 的執行效能，其不止搜尋相符的特徵值，並從已經學習與知道的資料庫中，也有特徵資料庫以幫助進來的事件使得表現更好。因此可以同時降低 False Positives 與 False Negative。在不可靠系統與可靠系統沒有直接連結，如同封包由代理伺服器的 IDS 詳細檢查。

表 2.6 簡化輸入變數之效能比較

Type of attack	Classification Accuracy %	
	EFuNN	ANN
Normal	99.56	99.57
Probe	99.88	94.62
DOS	98.99	98.97
U2R	65.00	59.00
R2L	97.26	97.02

EFuNN(Evolving Fuzzy Neural Network) 使用混合學習技

巧(混合 Unsupervised 與 Supervised Learning) 去調整 FIS 的參數，如同 EfuNN 調整一個單一的訓練(1 epoch)，更具調整性且容易進一步線上訓練，於較常使用的線上偵測與更新知識庫上更有用，故 EFuNN 更重要的特徵是使用者可以有彈性地建構網路。

本篇證明兩個機器學習設計 IDS，EFNN 效率較類神經網路好。實驗結果表示輸入簡化的重要，其輸入變數減少 40%，將可增加執行與發展的時間。

未來的 IDS 依靠資料相關性(Data Correlation)，明日的 IDS 將由不同的來源藉由檢查輸入以產生結果。此種將可解決統計分析與預測於人工智慧執行在異常的資料集合。

2.6 網路主機型入侵偵測系統

入侵偵測系統(IDS)主要目的是監控資源與在某個事件發生時通知某人，以進行適當的回應。以稽核資料的來源，IDS 可以分類為主機型入侵偵測系統 HBIDS 或網路型入侵測系統 NBIDS。

在本篇主要針對 NBIDS 與 IDS 的新概念稱為 NetHost-Sensor。本篇描述 NetHost-Sensor 的能力，以克服點對點加密(End-to-End Encryption)、阻斷式服務攻擊 DoS，與降低 False Positives。本篇的實驗程序與結果在設計 NetHost-Sensor。

入侵偵測系統(IDS)可以被定義為硬體/軟體系統，以監控發生在電腦與網路的事件，分析違反安全的情形，在本篇是指違反資料的完整性(Data Integrity)、機密(Configdentiality)與資源的的利用，其可相關參考資料[Abi03]。

Snort[Roe99]被使用 NBIDS 上，此系統為開放源碼(Open Source)並具有較輕量(Lightweight)效能的 IDS 工具，其可以部署在廣泛的各式平台。Snort 的特徵規則以登入與可以被執行的內容來搜尋與匹配(searching/matching)，可以被使用偵測各種攻擊與探測，例如：緩衝區溢位(Buffer Overflow), Stealth Port Scan, CGI (Common Gateway Interface)攻擊，SMB(Server Message Block)探測，OS fingerprinting 攻擊等。

雖然入侵偵測系統已經存在超過 20 年，在方法上有瑕疵

[Fre02][Kob][SWW00][Edit01]。

我們針對下列的弱點，提出方法改進：

1. End-to-End(ETE)加密：隨者在通訊的協議安全的改進，以點對點方式加密的情形越來越多。除阻撓駭客外，加密的內容使得 NBIDS 無法觀察其網路的封包內容與分析它的入侵行為，而造成高的 False Positive Rate。
2. 針對弱點直接攻擊：由於入侵偵測系統依靠階層式的結構，許多的 IDS 易受到攻擊。藉由攻擊內部的節點，可以切開 IDS 的分支，甚至由帶出去的 Root 命令與控制節點，把整個 IDS 切斷。一般來說，這樣的重要的元件存在平台上，要攻擊非常困難。因此，其他技術也缺乏備份(Redundancy), 移動性(Mobility), 動態回復(Dynamic Recovery) 等，其相關內容可參考[PB98]。

本篇主要探討在 DoS 攻擊的弱點上[Ove99]，針對監控的應用，透過 IPSec (Internet Protocol Security)[Dun01]，有 ETE (End to End) 加密攻擊。此弱點的解決方法，本篇提出“NetHost-Sensor”，藉由目標系統的通訊協定堆疊，以監控

造成目標系統應用上癱瘓的網路流量與不正常的系統呼叫。

2.6.1 其他網路型入侵偵測介紹

以下針對其他的網路型入侵偵測(NBIDS)研究進行介紹。

1. 實際安全網路偵測器 RNS (RealSecure Network Sensor)[ISS02]是由網際網路系統 ISS (Internet Security System)的入侵偵測產品。RNS 提供一個安全的分散架構，與使用多個偵測引擎，以進行入侵偵測。RNS 引擎可以被分散到各種網路，且將報告送回集中的管理主控台，另外引擎可以被設定偵測多種入侵，而引擎間的入侵報告與管理主控台以 128 bit RSA 加密。當入侵偵測引擎偵測到可疑的活動，即通知管理主控台，而主控台亦可以被啟動提供使用者 ODBC(Open Databas Connectivity)活動的紀錄，並以電子郵件通知管理者。

   RNS 之進階功能，如同進一步的通訊協定分析、過濾器、使用者特徵定義等，但其未提出 IP 切割攻擊(IP Fragmentation Attacks), 新增攻擊(Insertion Attacks)

與躲避攻擊(Evasion Attacks)[PN98]。因 RNS 被放置於目標系統與攻擊系統中間，故面臨點對點加密的異常行為的攻擊，且 RNS 偵測引擎無法解開加密的封包內容。

ISS 微小代理人(Micro Agents)[ISS]結合以稽核紀錄為基礎的代理人(Log-based Agent)，藉由微小代理人可觀看解密的封包，且不會遭受躲避攻擊。在早期 ISS 微小代理人的發展與 NetHost-Sensor 有一些重疊。

2. 由愛達荷州立大學電腦科學系[Ida02]提出使用防火牆移動式代理人(Firewall Mobile Agents)進行管理 VPN(Virtual Private Network)。如果使用者 B 想安全地與防火牆後面的主機 A 溝通，防火牆將送 FMCA(Firewall Mobile Custom Agent)給使用者 B 進行監偵到主機 A 的網路封包。FMCA 負責監測每一個網路封包，並將加密與簽章傳送給主機 A 的防火牆，若合法則靜態代理人在主機 A 防火牆確認簽章並允許通過。
3. 由 Vern Paxson 為避免某種形式的躲避攻擊[8]，以處理即時的分析 threads，對於每一個可能的模糊網路封包

(Ambiguous Network Packets)判斷。

4. 由 Thomas E. Daniels 與 Eugen H. Spafford[DS99]，嘗試偵測低階網路攻擊(Low-level Network Attacks)藉由目標系統的 Linux Kernel 紀錄為基礎。因此研究係針對使用者的通訊協定進行設計，故不能偵測點對點(ETE)加密與躲藏攻擊。而 NetHost-Sensor 有通訊協定分析技術，將可抵抗 ETE 加密與躲藏攻擊。

另外其他相關網路型入侵偵測系統的論文有 NetRanger[Abi03]，Netwok Flight Recorder 與 Peter Mell 等抵抗 DoS 之 IDS。

2.6.2 實驗說明與結果

實驗中包含 20 種 DoS 入侵攻擊，例如：W32.DoS 與 IGMP Nuke。這些攻擊由攻擊系統發動如下圖，Snort 分析網路封包，以偵測介於攻擊系統與目標系統間的加密通道入侵行為。

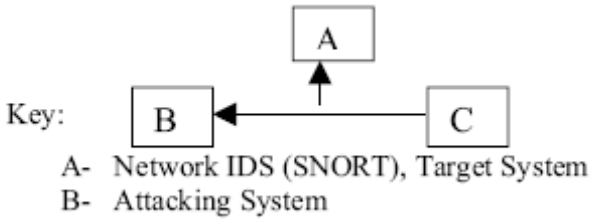


圖 2.11 Snort 分析網路封包架構

詳細的實驗如下：

(1)目的系統(Target System): Windows 2000

(2)網路入侵偵測系統(Network IDS): Windows 2000, 安裝 Snort 與 IRIS 網路分析器 (http://www.sunbelt-software.com)。

(3)攻擊系統(Attacking System): Windows 2000

(4)攻擊工具: 使用 25 種不同的工具，例如：由 DoS 工具、躲藏攻擊工具(Evasion Attack Tools)、通訊埠弱點攻擊(Port Vulnerability Attack Tools)。

實驗結果為測試 IPSec 點對點加密設定，本篇攻擊系統 Ping 到目標系統的 IP 位址。圖 2, 3 為網路抓取的資料內容。

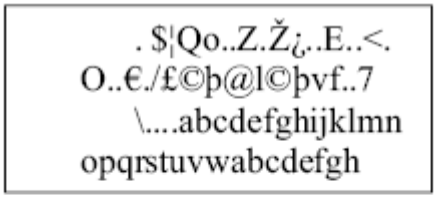


圖 2.12 Ping 正常封包的內容

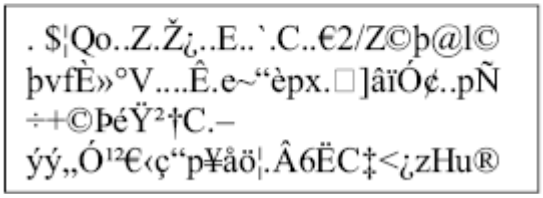


圖 2.13 Ping 加密封包的內容

在實驗的許多攻擊中，包含 DoS 攻擊與加密攻擊，其發動攻擊表示在下表。

表 2.7 DoS 攻擊與加密攻擊實驗結果

Attack Type Launched	Number of Attacks Launched	Attacks Detected
DoS	20	10
Encrypted	20	0

本篇發展 NetHost-Sensor 係針對加密的攻擊、躲藏攻擊、及利用程式弱點產生的 DoS 攻擊與 False Positives 攻擊。本篇未針對低階的 TCP/IP 層紀錄，而企圖使用應用層的資料進行偵測。Ptacek 與 Newsham[PN98]針對 NBIDS 提出避免陷阱之觀

點。

2.6.3 網路主機型入侵偵測技術

本節主要描述特徵值搜尋，大部分的偵測技術使用日期與定義新的偵測技術，稱為 NetHost-Sensor 通訊協定分析。

Snort 規則如下例說明：

Alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: “PHF; flags: AP; content: “/cgi-bin/phf”)

這規則將發生於 Snort，當一個 HTTP(Hyper-Text-Transfer Protocol) 要求由 URL(Universal Request Location) “/cgi-bin/phf” 發出。入侵者利用執行 PHF，想以特別的 URL 欺騙進而侵入系統，但藉由檢查所有進入的 URL 系統可以發現入侵者。在 Snort 中類似上項的規則有 500 條，其利用網路的封包內容(Packet Payload)比對所有的特徵值，而此種方法於搜尋特徵值時有很多問題，因未真實了解網路封包，使得特徵值 “/cgi-bin/phf” 於某種情形發生時，並非攻擊行為，而造成 NBIDS 發生 False Positive。

為進一步處理，NetHost-Sensor 將更深地了解網路封包的內

容，重新架構原來的資料意義。這需要更多的碼被寫下。例如 NetHost-Sensor 基本必須製作一個 Web 伺服器以偵測 HTTP 連結的特徵植。相同地，NetHost-Sensor 必須製作 FTP(File Transfer Protocol)伺服器，以偵測 FTP 入侵。

以下的文字表示典型的 HTTP Request:

GET/index.htmlHTTP/1.0Host:www.robertgrapham.comReferehttp://www.robertgraham.com/cgi-bin/phfUser-Agent:Mozilla/2.0

NetHost-Sensor protocol 使用更智慧的分析，將每一欄位分開，並給予表頭與欄位賦予有意義內容。

Method = GET

URL = /index.html

Version = HTTP/1.0

Fieldname = Host

HTTP_HOST = www.robertgraham.com

Fieldname = Referer

HTTP_FEFERER = http://www.robertgraham.com/cgi-bin/phf

Fieldname = User-Agent

HTTP8_USERAGENT = Mozilla/2.0

這一個例子的結果有 False Positive。此例子是要求

“/index.html”。然而，它可以使網路間封包含字串“/cgi-bin/phf”。當尋找有敵意的 URL 時，NetHost-Sensor 將尋找 URL 欄位，而不會針對其他網路封包。

躲藏攻擊(Evasion attack)的實例，可以使用 SNMP(Simple Network Management Protocol)網路封包，一般特徵值可以尋找某人企圖利用 Windows NT System 上的使用者帳號攻擊，其原始 Snort 特徵值如下：

Alert udp!$HOME_NET any->$HOME_NET 161(msg:”NETBIOS_SNMP-NT-UserList”;content: “[2b 06 01 04 04 4d 01 02 19|”;)

然而，SNMP 允許在資料上 Padding，一些額外的 Padding，在網路封包將實際送出內容為：

2b80 06 80 01 80 04 80 01 80 4d 80 01 80 02 80 19

因為原始的特徵值已經被修改，因此 Snort 將不再觸發此攻擊。相反地，NetHost-Sensor 系統將可自動地 Unsmudge 資料到其他的格式，不論有多少的資料被 Padding 在資料上，均可正確地偵測入侵。

大部分的通訊協定允許編碼(Encoding) 將相似排序或 Smudging，如此將隱藏入侵的真實特徵。NetHost-Sensor 將透它的通訊協定分析技術自動地管理，但大部分特徵搜尋 NBIDS 將偵測入侵失敗。

2.6.4 阻撓利用程式弱點之阻斷式攻擊

系統維持運作是重要的，即使其遭受攻擊的情形下，系統仍可維持運作，即安全與錯誤容忍係以基本服務維持為首要，甚至當系統被滲透或失敗時，提供所有服務快速的復原。

在系統存活的基本服務，此需要有三個能力：抵抗能力(Resistance)、識別能力(Recoginition)、恢復能力(Recovery)。抵抗能力是系統抵擋攻擊的能力。識別能力是攻擊發生時偵測能力，與評估損害範圍與危害能力。系統存活保證，主要是在遭受攻擊時能維持基本的服務與資產、限制損害範圍、及將所有服務重新建立。

本篇發展 NetHost-Sensor 概念，使用多種 DoS 評估，在系統損

毀或停止前，以發出系統呼叫其方式描述如下：

假設:

DoS 利用程式的弱點 = EXEDS(Appll)

Where

EXEDoS = executionofDoSexploit

Appll = exploitedapplication e.g. telnet

Assuming that：

系統呼叫有關於 Appll = H

因為應用程式的執行是一連續的系統呼叫，則

Execution of Appll = $H_0 \ldots H_{end}$.

Where;

H_0 = initial system call.

H_{end} = 應用程式最後執行的系統呼叫

故可說

ExeDoS(Appll)

$H_0 \ldots H_{DoS/a} \ldots H_{DoS/b} \ldots H_{DoS/crash-1} \ldots H_{DoS/crash}$

Where;

H_0 = initial system call.

$H_{DoS/a}$ …$H_{DoS/a}$ …$H_{DoS/a}$ …$H_{DoS/b}$ = 導致系統不正常的系統呼叫

$H_{DoS/crash-1}$ = 應用系統毀損的倒數第二個系統呼叫

$H_{DoS/crash}$ 使應用系統毀損的系統呼叫

由上述的描述將可推論抵抗 DoS 利用程式弱點，將可利用監測 H 的總和與停止任何系列的 H_0 … $H_{DoS/a}$ …$H_{DoS/b}$ …$H_{DoS/crash-1}$，其數學的公式如下：

$$\sum_{H0<H<HDos/a-1} EXEDoS(Appll)$$

為整個消除 DoS 利用威脅，本篇將決定主程序(Parent Process)，在 $H_{DoS/a}$ …$H_{DoS/crash-1}$ 間，獨立於任何系統呼叫。由任何 H 序列上，已知的 Appll，可以中止 Appll 與任何進一步的利用。

雖然藉由現今的學術研究與相關合作團體的研究，已超過 20 年但未達到原始動機的目標，本篇以闡述現今的 IDS 方法的弱點，包含 IDS 或目標系統的存活，於遭受到攻擊時，有很高的 False Positive 的數目。

本篇描述各種 DoS 攻擊，使用程式弱點與點對點加密為 DoS 的例子，與 False Positives 攻擊。

提出 NetHostSendor，NBIDS 阻撓點對點加密與 DoS 利用程式的弱點攻擊。NetHost-Sensor 特性有兩個新的概念：

(1) NetHost-Sensor 位於應用層與傳輸層間，以分析封包的內容。

(2) NetHost-Sensor 檢查目標系統之系統呼叫。

2.7 模糊類神經入侵偵測系統

透過網際網路發動的遠端攻擊將影響遍布全球連接於網際網路上之所有可靠網路，故已大量增加潛在的損失。因為每一系統均有弱點，入侵偵測於研究領域上愈來愈重要。模糊類神經網路入侵偵測系統(Neuro-Fuzzy Intrusion Detection System ;簡稱 NFIDS)是屬於 Anomaly 入侵偵測系統，使用模糊邏輯與類神經網路，來偵測惡意活動的發生。本篇係描述 NFIDS 架構與元件。

在最近幾年入侵偵測已有相當的成長，依照不同的需要已

經發展了，大量的入侵偵測系統。本篇作者提出新的入侵偵測定義，其為識別電腦系統誰有使用的權限，可合法存取系統，而拒絕非法使用者的權利[BIS98][GD01]。入侵偵測的技術可以被分為兩個互補的方法，Misuse 偵測與 Anomaly 偵測。Misuse 偵測以已知的入侵特徵與專家的報告為基礎，識別企圖入侵行為，或以弱點偵測誰使用新的特徵行為或誰偽裝合法行為以欺騙偵測系統。而 Anomaly 入侵偵測，則表達正常行為的特徵行為，假設入侵行為可以正常行為基礎，來識別入侵行為[BV00][ZLM01]。

入侵偵測系統可被分類為主機型或網路型。主機型系統其決定資訊的獲得係由單一台主機所提供。網路型的系統的資料，則係藉由監控網路上連結主機的追蹤資訊為基礎。IDS 的定義並未包含避免入侵的發生，只是偵測與入侵報告傳送給管理者。

模糊邏輯適合入侵偵測有兩個主要理由[BV00] [DJK01] [MW01]。第一，因有許多定量分析的特性包含在入侵偵測，例如不同的 TCP/UDP 服務的數目，藉由相同來源的主機可以被視為模糊變數。第二，使用模糊邏輯處理入侵偵測問題，其安全

包含模糊化，並給予定量分析方法，區間可以定義正常值，則任何值落入區間外，將被視為異常行為。使用模糊化表示這些定量的特徵值，以順利地分別正常與異常，並特別衡量正常與不正常的程度。模糊系統已經被使用在一些入侵偵測系統[GD01] [DD00] [DJK01]，模糊規則被表示已偵測每一型態的攻擊。類神經網路架構則提供一個執行模糊邏輯計算，在平行處理的結合表示。

2.7.1 入侵偵測架構

NFIDS 是屬於 Anomaly 入侵偵測系統，以自主的代理人架構[BIS98]與模糊類神經引擎偵測攻擊。系統存在三個階層[ZLM01]，不同的階層相對應有不同網路的範圍，透過代理人彼此溝通。第一層包含多個入侵偵測代理人(Intrusion Detection Agents; IDAs)。IDAs 是 IDS 的元件，監控主機或網路與第二層不正常的行為報告。第二層的代理人偵測區域網路(LAN)的狀態，以網路的流量可以觀察由第一層代理人與 LAN 間

的報告。第三層則為高階的報告，相關的資料與傳送警告到使用者的介面。系統的階層結構如下圖：

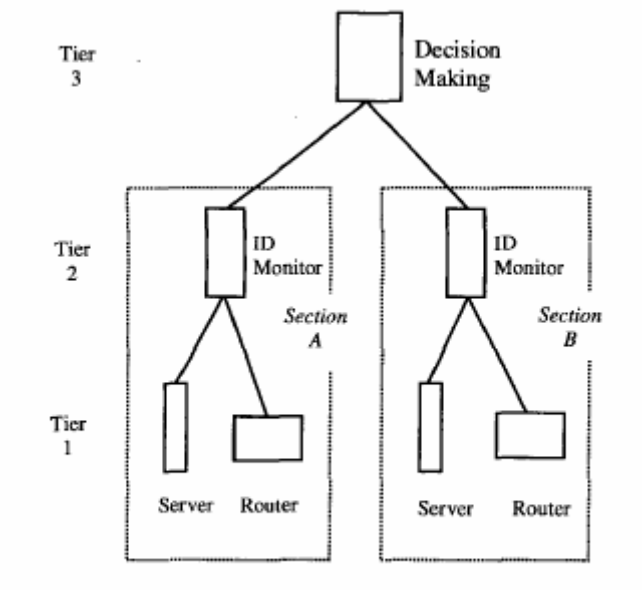


圖 2.14 模糊類神經入侵偵測系統

本系統有四種主要代理人的型態，說明如下：

(1) TCPAgent： 監控 TCP 介於主機與網路的連線。

(2) UDPAgent：尋找不正常的流量包含 UDP 資料。

(3) ICMPAgent：監控 ICMP 流量。

(4) PortAgent：尋找在網路上不正常的服務。

決策模組的主要架構，利用類神經網路與模糊邏輯偵測入侵。其架構圖形如下：

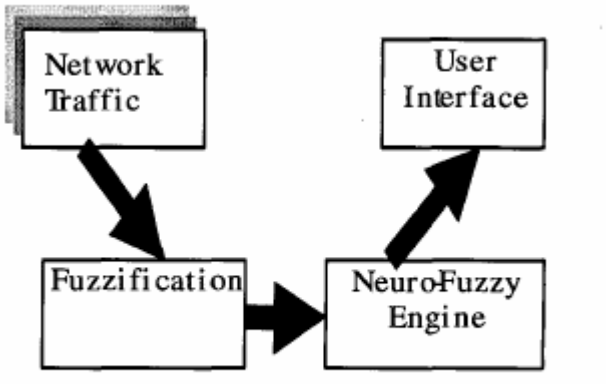


圖 2.15 決策模組架構

2.7.2 模糊類神經引擎

模糊系統的動態行為是藉由一組語言學上的規則，以描述特徵：

R_i : if (x_1 is A_{i1}) and … and (x_n is A_{in})

Then (y is B_l)

其 x 是模糊系統的輸入，y 是輸出[XM98]。使用上述模糊規則有很多不同的推理方法，其中有一個方法包含真實值的限制，命題“x is A” 比較 “x is A” 與其修正 B 的隸屬函數以獲得 B’ [XM98]。

模糊真值表限制 T 是模糊子集 X = [0, 1]，且定義隸屬函數 M, 及其對應:

M: X -> [0, 1]

在真值表限制方法，其給予實際的變數 A 值，事先機率的命題：If X is A then Y is B” 表示在真值表模糊子集合，真實區(Truth Space)的模糊子集合被使用在模糊簡化程序，以決定相對應的限制在真值(True-value)的命題“Y is B”。

表示命題“Y is B”,藉由前面提過的推理程序，$M_{B'}(v) = M_T(M_B(v))$。

類神經網路架構提供執行模糊邏輯計算，在 MLP(Multi-Layer Perceptron) 下圖，是多階層回饋型網路(Mulitplayer Feedforward Network)，包含輸入層、一個或多個隱藏層、與一個輸出層。MLP 可成功地解決各樣的困難[BMP02]，有其他作者進行 MLP 網路於多個隱藏神經元由 8 到 16 測試[ZM98].

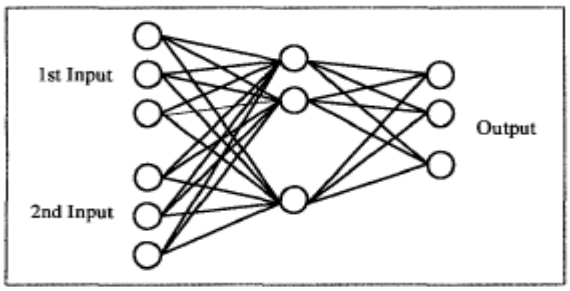


圖 2.16 多階層回饋型網路之模糊推理

模糊規則可以被學習，藉由這些網路與類神經網路可以執行模糊推理程序。

2.7.3 模糊規則

在設計系統架構的下一步驟，是定義模糊規則以偵測每一種型態的攻擊，這些規則可以使用專家知識。有一些輸入變數可以被使用在一些型態的攻擊：

1. NSD:表示數個連結介於來源與目的地的某個通訊埠連結。

2. USD:表示唯一連結於來源與目的地在特別的通訊埠。

另外使用下列五種語意名詞：

{LOW, MIDLOW, MID, MIDHIGH, HIGH}

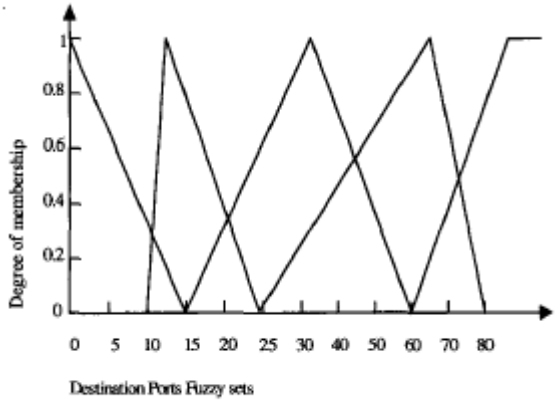


圖 2.17 屬性的模糊空間

於網路 Stealthy Port Scan 係入侵者送封包給多個已知的通訊埠，以尋找可以執行這些服務的系統[DD00]，於系統上存在這些服務，即可給予提示何種弱點攻擊，攻擊者將可嚐試利用以侵入系統。另外，攻擊者可以使用掃描識別在特定機器上的作業系統，藉由檢查 TCP 堆疊欺騙 TCP 控制訊息。正在執行的服務知識，與主機作業系統是非常有價值，因為可以幫助攻擊者縮小範圍，以找出系統的弱點並加以攻擊系統。通訊埠與作業系統偵測掃描可以是很重要的指標，使得可以找出更多嚴重的攻擊。輸出變數，決定輸入與輸出間的關係，這些型態的攻擊如下表：

表 2.8 網路掃瞄的模糊規則

Usd/ Nsd	L	ML	M	MH	H
L	**L**		**L**		
ML		**ML**			
M	**ML**				**MH**
MH					
H			**ML**		**H**

2.7.4 實驗結果

NFIDS 測試執行在 Tehran 的實際網際網路流量資訊與統計，其

測試資料蒐集 15 天。下表定義在模糊規則其結果如下：

表 2.9 模糊類神經網路實驗結果

Correct Normal Predictions	False Negatives	Correct Attack Predictions	False Positives
96%	4%	90.6%	9.4%

使用模糊邏輯，我們可以架構 if then 規則來表達一般方式描述安全攻擊。本研究有多種網路入侵行為的類別，其偵測能力優於以一般 Anomaly 基礎之模糊類神經系統，而一般的模糊入侵偵測掌握高階的入侵偵測方法並結合 Misuse 偵測系統。

第 3 章、應用防火牆紀錄探勘使用者行為

政府為加強為民服務品質與效率，業已將所有政府機構納入“e-Government”系統，於網際網路上提供全方位的民眾服務管道，此系統亦稱為政府服務網(Government Servive Network;簡稱 GSN)。因網際網路存在不安全因素例如偽造、竊聽、駭客入侵等，GSN 為加強網路安全另成立網路安全小組簡稱為 GSN CERT。

一般網際網路安全議題[CK00]可歸類為三部分：機密性(Secrecy)、完整性(Integrity)與可用性(Availability)，其詳細內容說明如下：

1. 機密性(Secrecy)：

   可進行資料的授權與未授權的控管，以達到電腦安全處理之資訊保護。

2. 完整性(Integrity)

   完整性概念係指資料的有效性與確保資料於傳輸時未曾被改變。

3. 可用性(Availability)

保證只有合法使用者可以使用系統資源。

入侵偵測係針對機密性與完整性進行管理,當有駭客入侵或潛在的威脅發生時,IDS 即傳送警訊通知管理者。一般入侵偵測系統可分為 Misuse 偵測與 Anomaly 偵測兩類。於本篇我們針對防火牆紀錄進行資料分析,利用前置碼基礎匹配方式(Prefix-Based Matching Scheme)將封包簡易分類,並搭配二元樹將來源 IP(Source IP)及目的地 IP(Destination IP)依前置碼資料簡化,將簡化過的防火牆紀錄以類神經 Fuzzy Art 網路方法進行分類,再依分類後之規則查對防火牆的規則(Policy)是否正確,借以適時調整及修正防火牆規則,及異常行為的偵測。

3.1 前置碼基礎匹配法(Prefix-based Matching Scheme)

在網際網路上,對於每一個流入至防火牆的封包均會檢查其規則表(Rule Table),依其規則表所預先定義之內容進行過濾,並且僅針對合法的封包進行轉送到目的地位址。因此可知防火牆係屬 IP 路由器(Router)的一種,而封包分類(Packet classification)是

為提升服務效能的一種技術，其技術係將進入的封包依預先定義的過濾規則分類為特定的類別，若封包同時符合多條規則時，則採用等級(priority)最高的規則。其中最長前置碼法(the Longest Prefix-matching Sheme) 普遍應用於使用封包表頭以尋找最合適的路由規則。

前置碼基礎匹配法亦屬一種封包分類方法，可利用相同的 IP 前置碼進行合併封包以簡化封包數量。若應用封包的多個欄位進行規則的比對，此種方式稱為多欄位封包分類(Multi-Field Packet Classification)，一般常使用的欄位有來源位址(Source Address)、目的地位址(Destination Address)、網路協定型態(Protocol Type)、來源埠(Source Port)、與目的埠(Destination Port)等。

本方法採用相同的規則與通訊協定下,將來源位址與目的地位址進行前置碼基礎匹配以簡化稽核紀錄之內容。例如：R0,udp,192.168.6.7,202.12.27.33 與 R0,udp,192.168.120.5,202.12.27.33(如表 1 防火牆稽核紀錄資料)，合併簡化為 R0,udp,192.168.*, 202.12.27.33。其中*表示任何數值,故 192.168.*

為前置碼 192.168 之所有 IP 位址。

表 3.1 防火牆稽核紀錄資料

Rule	Protocol	SA	DA
R2	udp	192.168.198.7	192.172.40.1
R2	udp	192.168.30.78	192.180.57.9
R3	udp	192.168.10.2	202.12.27.45
R5	tcp	192.168.0.2	210.69.91.66
R5	tcp	192.168. 0.2	210.69.91.77
R0	udp	192.168.6.7	202.12.27.33
R0	udp	192.168.120.5	202.12.27.33
R0	udp	192.168.192.10	202.12.27.33

表 3.2 前置碼基礎匹配簡化稽核紀錄

Rule	Protocol	SA	DA	Times
R0	udp	192.168.*	202.12.27.33	3
R2	udp	192.168.*	192.*	2
R3	udp	192.168.10.2	202.12.27.45	1
R5	tcp	192.168.0.2	210.69.91.*	2

3.2 防火牆規則檢核

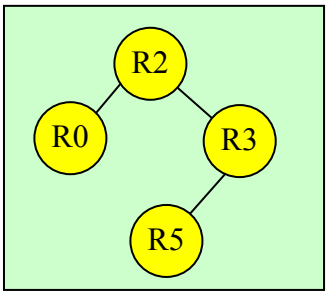


圖 3.1 前置碼二元樹

因防火牆預先定義之規則必須隨著網路環境變化而更動，且每一設定的更動均可能造成網路資訊安全缺口。故如何確保防火牆的規則定義是否正確，我們提出以防火牆的稽核紀錄資料檢視其防火牆規則設定的正確性。

因防火牆多為企業的網路安全入口設備，其連接外部的網際網路與內部的區域網路，所有進出網際網路的封包均須通過防火牆，故其每日稽核紀錄資料龐大，為簡化稽核紀錄利用前置碼基礎匹配

方式，將相同的前置碼 IP 合併為一筆稽核紀錄，最後再依合併後得到之結果進行分類，按其出現之頻率高低調整防火牆規則之檢查順序，其詳細步驟說明如下：

步驟 1. 依防火牆稽核紀錄資料建立前置碼二元樹

以 Rule, Protocol, SA, DA 為鍵值建立前置碼二元樹(Prefix Binary Tree)，建立二元樹過程中，若 Rule 與 Protocol 相同，則進一步比對 SA 與 DA 的前置碼，若其前置碼相同，則合併稽核紀錄之個數(Times)增加一，無須新增二元樹節點。例如：

R0,udp,192.168.6.7,202.12.27.33 與 R0,udp,192.168.120.5,202.12.27.33(如表 1 防火牆稽核紀錄資料)，合併簡化為 R0, udp,192.168.*,202.12.27.33。

步驟 2. 輸出前置碼基礎匹配簡化稽核紀錄

利用二元樹追蹤方式將簡化後的稽核紀錄輸出。

步驟 3. 依據前置碼基礎匹配簡化稽核紀錄，利用類神經 ART(Adaptive Reasonace Theory)再進一步分類簡化，並使用網路之出現頻率(次數)及分類檢討調整防火牆過濾

規則。

3.3 實驗結果

目前內政部土測量局之防火牆過濾規則計有 55 項，本實驗採用員工使用網際網路通過防火牆之紀錄 10,000 筆，經過前置碼基礎匹配法簡化稽核紀錄後，可簡化為 42 筆稽核記錄。茲將實驗結果說明如下：

1. 過濾規則使用頻率

表 3.3 防火牆規則使用頻率表

Rule	Times	Percent
R0	1756	3%
R1	437	1%
R14	223	0%
R15	38233	60%
R29	1643	3%
R32	14172	2%
R33	8206	13%
R45	349	1%
R47	4710	8%
R60	5358	9%

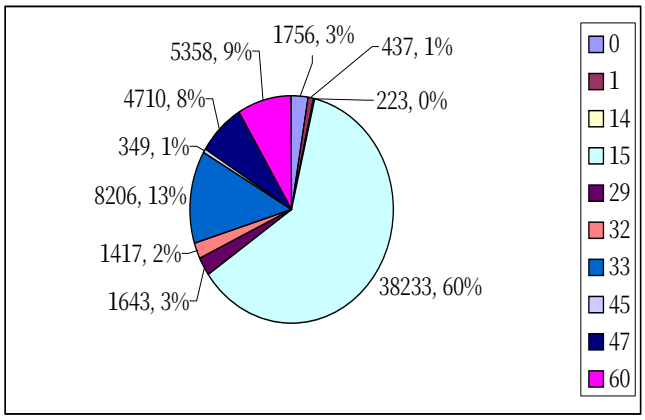


圖 3.2 防火牆規則使用頻率

說明：依據 94 年 8 月 6 日使用者通過防火牆的稽核記錄 64,818

筆資料中，所有防火牆規則計有 60 筆，而僅使用的過濾規則計有 22 類(R8, R21, R23, R28, R50, R52, R7, R9, R11, R44, R27, R24, R0, R1, R14, R15, R29, R32, R33, R45, R47, R60)，各規則的發生次數詳附錄，其中將發生次數較高者分別說明如下：

(1) R0: 電腦上網時查詢 DNS(Domain Name Server)。

(2) R1: AD Server 查詢。

(3) R14:本局員工及外業員工禁止使用 P2P 軟體(如：eDonkey, P2P 等)。

(4) R15:局本部員工可至任何目的位置進行所有服務。

(5) R29:任何來源可到防毒伺服器掃毒。

(6) R32:lsdms 對外開放 http 使用。

(7) R47:各測量隊及工作站可使用的各項服務(如：http, smtp, pop-3 等)。

(8) R60:防火牆規則最後一條規則,刪除(Drop)所有封包。

其使用頻率如表 3.4，其中使用 R15 的過濾規則最高，計有 38,233 次佔 60%，R33 次數 8,206 次佔 13%，R60 次數

5,358 次佔 9%，故應調整 R15, R33 順序至較高的過濾等級。

2.通訊協定使用頻率

經常使用之通訊協定有 TCP, UDP 及 ICMP，其出現的次數及百分比，TCP 共 4,165 次(78%)，UDP 共 1,195 次(22%)，及 ICMP 共 2 次(0%)。

表 3.4 通訊協定頻率表

Potocol	Times
ICMP	153
TCP	58,115
UDP	42,55

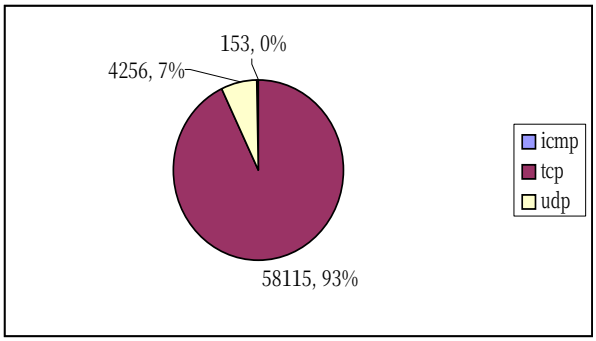


圖 3.3 通訊協定頻率

使用之通訊協定有 ICMP、TCP、UDP 等，以 TCP 所佔次數 58,115 次 93%最高，其原因為同仁使用 TCP 協定 http 上網瀏覽網頁或使用 Web 平台之業務系統越來越多所致。

第 4 章、結論及未來研究

現今面臨越來越複雜的網路環境，資訊全球化的快速進展，加上電子商務及國家公共服務應用，網路已實現地球村概念，但越來越多的弱點除可能導致電腦系統誤用與濫用外，網際網路上亦存在駭客或攻擊者，利用各種免費的掃描或駭客工具進行破壞，需要越來越多的資訊安全產品如：防火牆、防毒牆、弱點掃瞄軟體、入侵偵測系統等，以確保網路安全與保護資訊資產。

本項研究係針對內政部土地測量局防火牆稽核紀錄檔(Log)進行網路封包資料分析，實驗後發現防火牆第 15 及 33 項規則使用頻率最高，為增加防火牆效率將適時配合調整其規則；第 29 項防火牆規則為防毒軟體伺服器使用，但觀察其掃郵件(Smtp)數量相當少，其郵件遞送的路徑將進行檢討。另外，第 60 項規則為防火牆最後規則，亦即前 59 項規則均檢查不符合後即利用第 60 項規則丟棄該封包，依實驗顯示有 9%(5,358 筆)的封包需經所有規則檢查後，才發現為有問題封包再丟

棄，故將檢討是否應增加防火牆規則，以減少通過第 60 項規則發生的數量。

因防火牆為單位重要的管制通道，除加強管理規則設定及稽核外，未來將增加入侵偵測系統及無線網路方面研究其重點:

1. 針對無線網路封包及安全性進行入侵偵測。
2. 目前加密技術愈加成熟，如何使用加密技術於業務上，及加密環境的入侵偵測作業。
3. 使用 Mobile Agent 到網路上或主機上，主動資料蒐集，並傳送回入侵偵測系統，以判讀是否具有入侵行為。

參考文獻

[Abi03] Abiola Abimbola et. al, “Intrusion Prevention and Detectoin: A Survey”, Submitted to Journal of Computer Security, 2003.

[AFV95] D. Anderson, T. Frivold, and A. Valdes, “Next-Generation Intrusion Detection Expert System (NIDES) A Summary,” http://www.sd1.sri.com/paper/4/s/4sri/4sri.pdf, Computer Science Laboratory, SRI Intl, May 1995.

[BMP02] A. Bivens, M. Embrenchts, C. Palagiri, R. Smith, B.Szymanski, “Network-Based Intrusion Detection Using Neural Networks,” Artificial Neural Networks In Engineering, St. Louis, Missouri, Nov. 10-13, 2002.

[BJPW2001] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu, “ADAM:Detecting Intrusion by Data Mining,” Proc. 2001 IEEE Workshop Information Assurance and Security, pp. 11-16, June 2001.

[BIS98] J. S. Balasubramaniyan, J. O. Garcia-Fernandes, D. Isacoff, E. Spafford, D. Zamboniy, “An Architecture for Intrusion Detection using Autonomous Agents,” Purdue University CERIAS Tehinical Report 98/05, Center for Education and Research in Information Assurance and Security, June 11, 1998.

[BG89] R. A. Baeza-Yates and G. H. Gonnet, “A New Approach to Text

Searching," Proc. 12th Ann. ACM-SIGIR Conf. Information Retrieval, pp. 168-175, June 1989.

[BV00] S. Bridges, R. Vaughn, "Intrusion detection via fuzzy data mining," Proceedings of the 12th annual Canadian information technology security symposium, Communications Security Establishment, pp. 111-121, Ottawa, June 2000.

[CGK94] Calvin Ko, George Fink, and Karl Levitt, "Automated detection of vulnerabilities in priviledged programs by execution monitoring," In Proceedings of the 10th Annual Computer Security Applications Conference, pp. 134-144, 1994.

[CERT93] CERT. Wuarchive.ftpd vulnerability. ftp://info.cert.org/pub/cert_advisories/CA-93:53.wuarchive.ftpd.vulnerability, 1993.

[Cox94] E. Cox, "The Fuzzy Systems Handbook,," Academic Press, Inc., UK. 1994.

[CS95] M. Crosbie and G. Spafford, "Defending a computer system using autonomous agents," In Proceedings of the 18th National Information Security Systems Conference, 1995.

[Den87] D. E. Denning, "An intrusion detection model," IEEE Transactions on Software Engineering, Los Alamos, CA, 1987.

[Den92] D. E. Denning, "Cryptography and Data Security," Addison-Wesley Inc, 1992.

[DN85] D. E. Denning and P. G. Neumann , “Requirements and model for IDES-A real-time intrusion detection system,”Comput. Sci. Lab, SRI International, Menlo Park, CA, Tech. Rep., 1985.

[DJK01] J. E. Dickerson, Jukka Juslin, Ourania Koukousoula, Julie A. Dickerson, “Fuzzy Intrusion Detection,” Pro. Of 20th NAFIPS International Conference, Jul. 2001.

[DD00] J.E. Dickerson, J.A. Dickerson., “Fuzzy Network Profiling for Intrusion Detection,” PeachFuzz 2000. 19th International Conference of the North American Fuzzy Information Processing Society, pp. 301-6, 2000.

[DS99] T. Daniels and E.H. Spafford, “Identification of Host Audit Data to Detect Attacks on Low-level IP,” Computer Security, Vol. 7, No. 1, 1999.

[Edit01] Editorial:., “Active Networks and Services”, Computer Networks , Vol 36, pp. 1-3, 2001

[FV95] D. Farmer and W. Venema, “Improving the security of your site by breaking into it,” ftp://ftp.win.tue.nl/pub/security/admin-guide-to-cracking.101.Z, 1995.

[FS90] Daniel Farmer and Eugene H. Spafford, “The COPS security checker system,” In Proccedings of the Summer Usenix Conference, pp. 165-170, 1990.

[Fau94] Fausett L., “Fundamentals of Neural Networks,” Prentice Hall,

1994.

[FBV2002] German Florez, Susan M. Bridges, and Rayford B. Vaughn, “An Improved Algorithm for Fuzzy Data Minig for Intrusion Detection,” IEEE NAFIPS. Annual Meeting of the North American, June 2002.

[Fre02] Christopher Kruegel Fredrik., “Stateful Intrusion Detection for High-Speed Networks”, IEEE Symposium on Security and Privacy, pp. 12-15, 2002.

[FHS96] S. Forrest, S. A. Hofmeyr, and A. Somayaji, “A sense of self for unix processes,” In Proccedings of the 1996 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, 1996. IEEE Computer Society Press.

[GC98] A. K. Ghosh, J. Wanken, and F. Charron, “Detecting Anomalous and Unknown Intrusion Against Programs,” Proc. 1998 Computer Security Applications Conf., pp. 259-267, 1998.

[GD01] J. Gomez, D. Dagupta, “Evolving Fuzzy Classifiers for Intrusion Detection,” Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 5-6 June 2001.

[HJS93] J. Hochberg, K. Jackson, C. Stallings, J. F. McClary, D. DuBois, and J. Ford. Nadir, “An automated system for detecting network intrusion and misuse,” Computeres and Security, Vol 12, No. 3, pp. 235-248, 1993.

[HML92] L. T. Heberlein, B. Mukherjee, and K. N. Levitt, “Internet security monitor: An intrusion detection system for large scale networks,” In Proceedings of the 15th National Computer Security Conference, 1992.

[HDL90] L. T. Heberlein, G. V. Dias, K.N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, “A network security monitor,” In Proceedings of the IEEE Symposium on Security and Privacy, IEEE Press, 1990.

[HLMS90] R. Heady, G. Luger, A. Maccabe, and M. Servilla, “The architecture of a network level intrusion detection system Technical report,” Department of Computer Science, University of New Mexico, 1990.

[HFS98] Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji, “Intrusion Detection using Sequences of System Calls,” Computer Secutity, No. 6, pp.151-180, 1998.

[ISS02] Internet Security System , “Internet Security Systems Ships Most Advanced Network Intrusion Protection System”, Realsecure 7.0, http://www.iss.net/, June 2002.

[ISS] Internet Security System, “Intrusion Detection For the Millennium,”, http://documents.iss.net/whitepapers/int_detet.pdf. 2002.

[IKP95] K. Illgun, R. A. Kemmerer, and P. A. Porras , “State trasition analysis: A rule-based intrusion detection approach,” IEEE Transactions on Software Engineering, 3 1995.

[JV91] H. Javitz and A. Valdes, “The SRI IDES statistical anomaly detector,” in Proc. Symp. Research Security Privacy, Menlo Park, CA, 1991, pp. 316-326.

[KW98] C. Kuok, A. Fu, and M. Wong, “Mining fuzzy association rules in databases”, SIGMOD Record Vol. 1, No. 27, pp. 41-46, 1998.

[KFL94] Calvin Ko, George Fink, and Karl Levitt , “Automated detection of vulnerabilities in priviledged programs by execution monitoring,” In Proceedings of the 10th Annual Computer Security Applications Conference, pp.134-144, 1994.

[Kep94] J. O. Kephart, “A biologically inspired immune system for computers,” In Artificial Life IV. MIT Press, 1994.

[Kob] Jonathan Koba, “Windows NT Attacks for the valuation of Intrusion Detection Systems”, Master of Engineering in Electrical Engineering and Computer Science Thesis, Massachusetts Institute of Technology.

[Kas98] Kasabov N., “Evolving Fuzzy Neural Networks- Algorithms,” Applications and Biological Motivation, in Yamakawa T and Matsumoto G (Eds), Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, pp. 217-274, 1998.

[KDD99] KDD cup 99 Intrusio detection data set <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz > <http://www.ll.mit.edu/IST.ideval/>.

[KYM02] L. Kohout, A. Yasinsac, and E. McDuffie, “Activity Profiles for

Intrusion Detection," Proc. North Am. Fuzzy Information Processing Society-Fuzzy Login and the Internet (NAFIPS-FLINt 2002), June 2002.

[KS94] Sandeep Kumar and Eugene H. Spafford, "A pattern matching model for misuse intrusion detection," In Proceedings of the National Computer Security Conference, pp. 11-21, 1994.

[Kum95] Sandeep Kumar , "Classification and Detection of Computer Intrusions," PhD thesis, Department of Computer Sciences, Purdue University, 1995.

[LH01] Susan C. Lee and David V. Heinbuch, "Training a Neural-Network Based Intrusion Detctor to Recongnize Novel Attacks", IEEE Transactions on System, Man Cybernetics—Part A: Systems and Humans, Vol. 31, No. 4, pp.294-299 ,July 2001.

[Low96] G. Lowe, "Some New Attacks upon Security Protocols," Proc.Ninth IEEE Computer Security Foundations Workshop, pp.162-169, Mar. 1996.

[LV92] G. Liepins and H. Vaccaro, "Intrusion detection: Its role and validation," Computers and Security, No. 11. pp. 247-355, 1992.

[Lau02] Lau C. et al, "Intrusion Signatures and Analysis", SANS Giac, 2002 Reprint.

[LCP00] T. Lonjgstaff, C. Chittister, R. Pethia, "Are we forgetting the risks of information technology," IEEE Computer, pp. 43-51, Dec. 2000.

[Lau91] Lau C., “Neural Networks, Theoretical Foundations and Analysis”, IEEE Press, 1991.

[LJ88] T. Lunt and R. Jagannathan, “A Prototype Real-Time Intrusin Detection Expert System,” Proc. IEEE Symp. Security and Privacy, pp. 59-66, Apr. 1988.

[LTG92] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P.G. Neumann, H. S. Javitz, A. Valdes, and T. D. Garvey, “A real-time intrusion detection expert system (IDES)—final technical report,” Computer Science Laboratory, SRI International, Menlo Park, 1992.

[Lun93] T. F. Lunt., “Detecting intruders in computer systems,” In Conference on Auditing and Computer Technology, 1993.

[MW01] M. Manic, B. Wilamowski, “Fuzzy Preference Approach for computer Network Attack Detection ,” International Joint Conference on Neural Networks (IJCNN’01), pp. 1345-1349, Washington DC, July.15-19, 2001.

[MSA] Mukkamala S., Sung A.H. and Abraham A., “Intrusion Detection Using Ensemble of Soft Computing Paradigms,” Intelligent Systems Design and Applications, Abraham A., Koppen M. and Franke K. (Eds.), Springer Verlag, Germany, pp. 239-248, 2003.

[MIT] MIT Lincoln Laboratory. <http://www.ll.mit.edu/IST.ideval/>.

[MSA03] Mukkamala S., Sung A.H. and Abraham A., “Distributed Multi-Intelligent Agent Framework for Detection of Stealthy Probes,

Design and Application of Hybrid Intelligent Systems, Abraham A,, Koppen M. and Franke K. (Eds.), IOS Press," Amsterdam, The Netherlands, pp. 116-125, 2003.

[MHL94] Mukherjee B., Heberlein T.L. and Levitt K.N., "Network intrusion detection," IEEE Network, Vol 8, No. 3, 1994.

[MA75] Mamdani E.H. and Assilian S., "An experiment in Liguistic Synthesis tieh a Fuzzy Login Controller," International Journal of Man-Machine Studies. Vol. 7, No. 1, pp. 1-13, 1975.

[Man94] M. Manic, "Alarm systems for monitoring driven by fuzzy logic," Proceeding & Information Technologies, Nis, pp. 30.1-30.4, Decembe 1994.

[Ove99] Overill R. E, "Denial of service attacks: Threats and Methodologies", Journal of Financial Crime, Vol. 6, pp. 351-354, 1999.

[PN97] P. A. Porras and P. G. Neumann, "EMERALD: Event Monitoring Enableing Responses to Anomalous Live Disturbances," Proc. Nat'l Information Systems Security Conf., pp. 353-365, Oct. 1997.

[PB98] Paxson, V,. Bro "A System for Detecting Network Intruders in Real-Time", Proceeding of the 7th USENIX Security Symposium, San Antionio, Taxas, January 1998.

[PN98] Ptacek. T and Newsham T., "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, Secure Networks", Inc,. http://www. acri.org/vern/ptacek-NewshamEvasion-98, January 1998.

[Roe99] Martin Roesch., “Snort-Lightweight Intrusion Detectoin for Networks,” USENIX LISA Conference , 1999.

[SS96] C. L. Schuba and E. H. Spafford, “Countering abuse of name-based authentication,” In 22cnd Annual Telecommunications Policy Research Conference, 1996.

[Syv94] P. Syverson, “A Taxonomy of Replay Attacks,” Proc Seventh Computer Security Foundations Workshop, pp. 131-136, June 1994.

[SW94] S. E. Smaha and J. Winslow, “Misuse detection tools,”Journal of Computer Security, Vol 10, No.1 pp. 39-49, 1994.

[SWW00] Stolfo. S, Fan. W, Lee. W, et al., “Cost-Based Modelling for Fraud and Intrusion detection: Result from the Jam Project”, In Proceeding of the DARPA Infromation Survivability Conference and Exposition, pp. 130-144, IEEE Computer Press, 2000.

[TCL90] Henry S. Tseng, Kaihu Chen, and Stephen C. Lu., “Security audit trail analysis using inductively generated predictive rules,” In Proccedings of the Sixth Conference on Artificial Intelligence Applications, pp. 24-29, 1990. IEEE.

[TA04] Tysen Leckie, and Alec Yasinsac, “Metadata for Anomaly-Based Security Protocol Attack Deduction,” IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 9, 2004.

[WS96] D. Wagner and B. Schneier, “Analysis of SSL 3.0 Protocol,” Proc. Second USENIX Workshop Electronic Commerce, pp. 29-40, Nov.

1996.

[WF74] R. A. Wagner and M. J. Fischer, “The String-to-String Correction Problem,” J. ACM, Vol. 21, pp. 168-178, Jan. 1974.

[WM91] S. Wu and U. Manber, “Fast Text Searching With Errors,” Technical Report TR. 91-11, Detp. Of Computer Science, Univ. of Arizona, 1991.

[WL92] T.Y.C. Woo and S.S. Lam, “Authentication for Distributed Systems,” Computer, Vol.25, No. 1, pp.39-52, Jan. 1992.

[XM98] M. H. Xsnf, M.T. Mohsjrtsni, et. al., “On a Neuro-Fuzzy Controller for High Degrees of Freedom Robot Manipulators,” IEEE International Conference on Intelligent Engineering Systems, Vienna, Austria, September, 1998.

[Yas02] A. Yasinsac, “An Environment for Security Protocol Intrusion Detection,” Computer Security, Vol. 10, No. 1-2, pp. 177-188, 2002.

[YC01] N. Ye and Q. Chen, “An Anomaly Detection Tcchnique Based on a Chi-Square Statistic for Detecting Intrusion into Information System,” Proc. Quality and Reliability Eng. Int’l, Vol. 17, No. 2, pp. 105-112, 2001.

[Zad74] AL. Zadeh, “Appendix,” Proc. of the U.S.-Japan Seminar on Fuzzy Sets and Their Application. Berkeley, Ca., pp. 27-39, July 1974.

[ZM98] M. H. Zand, M. R. Mohajerani, et. al., “On a Neuro-Fuzzy Controller for High Degrees of Freedom Robot Manipulators,” IEEE

International Conference on Intelligent Engineering Systems, Vienna, Austria, September 1998.

[68] P. R. Stephenson, “Intrusion Management Paper: A Top level Model for Securing Information Assets in an Enterprise Environment,” Enterprise Networking Systems, Inc., 2000.

[LCP00] T. Lonjgstaff, C. Chittister, R. Pethia, “Are we forgetting the risks of information technology,” IEEE Computer, pp. 43-51, Dec. 2000.

[LP99] U. Lindqvist and P. A. Porras, “Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST),” Proc. 1999 IEEE Symp. Security and Privacy, pp. 146-161, May 1999.

[Ida02] University of Idaho, Department of Computer Scystem, Firewall Mobile Customs Agent Project, http://www.csuidaho.edu/%7Eabdullah/fmca/index.htm, 2002.

[HK01] V. Hallivuori and M. Kousa, “Denial of Service Attack against SSH Key Exchange,” Telcomm. Software and Multimedia Laboratory, Helsinki Univ. of Technology, Nov 2001.

[LS98] W. Lee and S.J. Stolfo, “Data Mining Approaches for Intrusion Detection,” Proc. Seventh USENIX Security Symp., pp.26-29, Jan, 1998.

[LS00] W. Lee and S. J. Stolfo, “A Framework for Constructing Features and Models for Intrusion Detection Systems,” Proc. ACM Trans.

Information and System Security, pp. 227-261, Nov. 2000.

[LP99] U. Lindqvist and P. A. Porras, “Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST),” Proc. 1999 IEEE CS Symp. Security and Privacy, pp. 146-61, 1999.

[ZLM01] Z. Zhang, J. Li, C. N. Manikopoulos, J. Jorgeneson, J. Ucles, “HIDE: a Hierachical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification,” Proceedings of 2001 IEEE Man Systems and cybernetics Information Assurance Workshop, 2001.

[LGM94] [8LGM]. [81gm]-advisory-16.unix.sendmail-6-dec-1994. http://www.8lgm.org/advisories.html.

[LGM95] [8LGM]. [81gm]-advisory-22.unix.syslog.2-aug-1995. http://www.8lgm.org/advisories.html.

[LGM91] [8LGM]. [81gm]-advisory-3.unix.lpr.19-aug-1991. http://www.8lgm.org/advisories.html.

[CK00] A. Chaudhury, J. P. Kuilboer, “e-business and e-commerce infrastructure: Technologies supporting the e-business initiative,” George Werthman, 2000.

.

附錄 1：

使用之過濾規則

Rule	Times
8	1
21	1
23	1
28	1
50	1
52	1
7	9
9	12
11	20
44	24
27	39
24	84
0	1756
1	437
14	223
15	38233
29	1643
32	1417
33	8206
45	349
47	4710
60	5358

前置碼基礎匹配法簡化稽核紀錄

Rule	Protocol	SA	DA	Times
0	udp	12*	210.69.91.65	4
0	udp	128.174.5.58	210.69.91.65	1
0	udp	130.132.50.10	210.69.91.65	1
0	udp	139.175.55*	210.69.91.65	7
0	udp	140*	210.69.91.65	13
0	udp	147.208.15.174	210.69.91.65	1
0	udp	148.244.150.20	210.69.91.65	1
0	udp	159.18.144.10	210.69.91.65	1
0	udp	163*	210.69.91.65	5
0	udp	166.102.165*	210.69.91.65	3
0	udp	168*	210.69.91.65	381
0	udp	172.10*	192*	713
0	udp	172.10*	168.95.1.1	22
0	udp	192.168*	140*	63
0	udp	192.168.0.2	198.41.0.4	1
0	udp	192.168.0*	139*	2
0	udp	192.168.0*	69*	4
0	udp	192.168*	168.95*	51
0	udp	192.168*	192*	98
0	udp	192.168.0*	218*	4
0	udp	192.168.0*	211*	5
0	udp	192.168.0*	66*	21
0	udp	192.168.0*	61*	20
0	udp	192.168.0*	194*	3
0	udp	192.168.0.2	204.74.112.1	1
0	udp	192*	210*	62
0	udp	192.168.0*	202*	13
0	udp	192.168.0*	216*	3
0	udp	192.168.0.2	205.209.133.30	1

Rule	Protocol	SA	DA	Times
0	udp	192.168.0*	60.248.22*	4
0	udp	192.168.0.2	219.136.252.93	1
0	udp	192.168.0*	207*	13
0	udp	192.168.0.23	206.132.100.108	1
0	udp	192.168.0.23	203*	6
0	udp	192.168.0.23	65.39.176.4	1
0	udp	192.168.0.23	220.130*	2
0	udp	192.168.0.23	213.36.80.2	1
0	udp	192.168.0.23	80.237.244.2	1
0	udp	192.168.0.23	152.163.159.234	1
0	udp	192.168.0.23	193.70.192.100	1
0	udp	192.168.0.23	64.255.161.55	1
0	udp	192.168.0.23	217.75.120.123	1
0	udp	192.168.0.23	163.28*	2
0	udp	192.168.0.23	208.185.132.166	1
0	udp	192.168.0.23	63.105.207.85	1
0	udp	192.168.0.23	67.15.35.167	1
0	udp	193.219.214.10	210.69.91.65	1
0	udp	194.167.55.119	210.69.91.65	1
0	udp	195.42.198.10	210.69.91.65	1
0	udp	196.2.45.101	210.69.91.65	1
0	udp	199.199.208.2	210.69.91.65	1
0	udp	200*	210.69.91.65	4
0	udp	202*	210.69.91.65	13
0	udp	203*	210.69.91.65	16
0	udp	205.134.161.204	210.69.91.65	1
0	udp	206*	210.69.91.65	2
0	udp	207.218.192*	210.69.91.65	2
0	udp	208*	210.69.91.65	2
0	udp	209.200.18.100	210.69.91.65	1
0	udp	210*	210.69.91.65	43

Rule	Protocol	SA	DA	Times
0	udp	210.69.91.190	168.95.1.1	1
0	udp	211*	210.69.91.65	35
0	udp	212.242.32.226	210.69.91.65	1
0	udp	213*	210.69.91.65	2
0	udp	216*	210.69.91.65	14
0	udp	217.237.151*	210.69.91.65	4
0	udp	218.25.41.136	210.69.91.1	1
0	udp	220.135.30.15	210.69.91.65	1
0	udp	221*	210.69.91.65	2
0	udp	24.25.4.106	210.69.91.65	1
0	udp	38.118.142.3	210.69.91.65	1
0	udp	4.78.22.9	210.69.91.65	1
0	udp	57.79.9.118	210.69.91.65	1
0	udp	61*	210.69.91.65	11
0	udp	63*	210.69.91.65	3
0	udp	64*	210.69.91.65	9
0	udp	65*	210.69.91.65	5
0	udp	66*	210.69.91.65	9
0	udp	67.19*	210.69.91.65	9
0	udp	68.142*	210.69.91.65	3
0	udp	69*	210.69.91.65	3
0	udp	70.84.160*	210.69.91.65	5
0	udp	72.36.202.218	210.69.91.65	1
1	udp	192.168*	192.168.0.23	102
1	udp	172.10*	192.168.0.23	335
7	tcp	210*	210.69.91.190	6
7	tcp	80.18.163.179	210.69.91.190	1
7	tcp	220.174.91.144	210.69.91.190	1
7	icmp	24.87.43.41	210.69.91.190	1
8	tcp	192.168.0.23	67.15.35.167	1
9	tcp	210*	210.69.91.251	5

Rule	Protocol	SA	DA	Times
9	udp	220.241.211.92	210.69.91.251	1
9	tcp	80.18.163.179	210.69.91.251	1
9	tcp	220.174.91.144	210.69.91.251	1
9	tcp	221.164.87.99	210.69.91.251	1
9	icmp	210.132.180.104	210.69.91.251	1
9	icmp	218*	210.69.91.251	2
11	tcp	172.10*	192.5.6.32	19
11	tcp	210.69.91.190	168.95.1.1	1
14	tcp	192.168.140.59	211.144.62.23	1
14	tcp	192.168.140.59	219.71.95.107	1
14	tcp	192.168.150.120	201.130.67.50	1
14	udp	192.168.200.50	222*	2
14	udp	192.168.200.50	85.144.76.19	1
14	udp	192.168.200.50	68.155.197.82	1
14	tcp	172.10.71.21	81.216.110.47	1
14	tcp	192.168.150.120	62.219.164.144	1
14	tcp	192.168*	207.46*	211
14	tcp	192.168*	65.54.239.210	3
15	tcp	192.168*	207*	560
15	tcp	192.168*	61*	4037
15	tcp	192.168*	209*	153
15	tcp	192.168*	220*	1355
15	udp	192.168*	140*	207
15	tcp	192.168*	140*	720
15	tcp	192.168*	64*	490
15	tcp	192.168*	12*	15
15	udp	192.168.140.38	159*	2
15	udp	192.168*	81*	12
15	udp	192.168.140*	153*	3
15	udp	192.168*	62*	24
15	udp	192.168*	24*	39

Rule	Protocol	SA	DA	Times
15	udp	192.168*	200*	11
15	udp	192.168.140*	201*	2
15	udp	192.168*	65*	2
15	udp	192.168*	67*	11
15	tcp	192.168*	218*	2889
15	udp	192.168*	69*	88
15	udp	192.168.140.38	129.186.109.227	1
15	udp	192.168*	68*	20
15	udp	192.168*	202*	50
15	udp	192.168.150.101	138.37.135.68	1
15	udp	192.168*	61*	143
15	udp	192.168*	213*	72
15	udp	192.168*	130*	3
15	udp	192.168*	83*	9
15	udp	192.168*	219*	76
15	udp	192.168.150.101	72*	2
15	udp	192.168*	220*	273
15	udp	192.168*	84*	17
15	udp	192.168*	70*	35
15	udp	192.168*	85*	8
15	udp	192.168*	147*	9
15	udp	192.168*	210*	69
15	udp	192.168.140.8	136.167.172.130	1
15	udp	192.168*	82*	37
15	udp	192.168*	80*	9
15	udp	192.168*	60*	3
15	udp	192.168*	66*	2
15	tcp	192.168*	203*	5780
15	udp	192.168.140.38	152.2.57.149	1
15	udp	192.168.140.16	195.116.231.21	1
15	udp	192.168*	203*	9

Rule	Protocol	SA	DA	Times
15	udp	192.168*	211*	9
15	tcp	192.168*	59*	907
15	tcp	192.168*	219*	1136
15	udp	192.168.150.101	209.213.245.124	1
15	udp	192.168.180.11	134.147.66.123	1
15	udp	192.168.150.101	143.89.57.68	1
15	udp	192.168*	217*	7
15	udp	192.168*	218*	38
15	tcp	192.168*	69*	421
15	tcp	192.168*	72.9*	80
15	udp	192.168.180.11	165.123.178.161	1
15	udp	192.168*	221*	47
15	tcp	192.168*	83*	657
15	udp	192.168.150.101	156.17.224.139	1
15	tcp	192.168*	217*	50
15	udp	192.168*	205*	2
15	udp	192.168*	194*	28
15	tcp	192.168*	210*	3253
15	udp	192.168.140.38	128*	3
15	udp	192.168.140.38	212.76.47.63	1
15	tcp	192.168*	202*	6628
15	udp	192.168*	59*	10
15	tcp	192.168*	211*	2510
15	tcp	192.168*	60*	74
15	udp	192.168*	222*	6
15	tcp	192.168*	70*	431
15	tcp	192.168*	58*	32
15	tcp	192.168*	81*	13
15	tcp	192.168*	68*	71
15	tcp	192.168*	67*	155
15	tcp	192.168*	195*	276

Rule	Protocol	SA	DA	Times
15	tcp	192.168*	163*	298
15	udp	192.168*	64.4.12*	7
15	tcp	192.168*	192*	362
15	tcp	192.168.150.120	82.82.103.144	1
15	tcp	192.168*	71*	2
15	tcp	192.168*	221*	80
15	tcp	192.168*	222*	332
15	tcp	192.168*	24*	121
15	udp	192.168.0.21	255.255.255.255	1
15	tcp	192.168*	205*	44
15	udp	192.168.140*	163*	24
15	udp	192.168.150.101	132.230.167.34	1
15	tcp	192.168.150.120	84.248.43.213	1
15	tcp	192.168.150.120	200.93.76.190	1
15	tcp	192.168.150.120	151.202.54.45	1
15	tcp	192.168.140.59	201.9.20.49	1
15	udp	192.168*	137*	2
15	udp	192.168.150.120	216.156.142.13	1
15	udp	192.168.150.120	207.226.112.22	1
15	udp	192.168.150.120	63.217.27.11	1
15	udp	192.168.150.101	18.34.1.115	1
15	udp	192.168.140.38	150.164.72.62	1
15	tcp	192.168*	213*	4
15	tcp	192.168*	66*	338
15	tcp	192.168*	63*	46
15	tcp	192.168*	65*	349
15	tcp	192.168*	208*	87
15	tcp	192.168*	216*	396
15	tcp	192.168*	206*	7
15	tcp	192.168*	193*	90
15	tcp	192.168*	204*	38

Rule	Protocol	SA	DA	Times
15	tcp	192.168.120.1	198.65.220.236	1
15	tcp	192.168*	164.109.43.3	2
15	tcp	192.168.140.16	131.107.115.28	1
15	tcp	192.168*	212*	32
15	tcp	192.168*	172*	507
15	tcp	192.168*	139.175*	46
15	tcp	192.168.140.50	170.224.8.51	1
15	tcp	192.168*	165*	4
15	tcp	192.168*	152*	8
15	tcp	192.168*	38*	199
15	tcp	192.168.140.7	146.82*	89
15	tcp	192.168*	168*	119
15	tcp	192.168.150*	80.237.244*	9
15	tcp	192.168.150.120	130.117.152.163	1
15	tcp	192.168.150.120	150.65.7.130	1
15	tcp	192.168*	129*	29
15	tcp	192.168.180.62	62.190.189.158	1
15	tcp	192.168.200.3	128.30.52.25	1
15	tcp	192.168.0.1	4.79.181*	2
15	udp	192.168.0.241	192.168*	402
15	tcp	192.168.0.250	10.100.100.118	1
15	icmp	192.168.0.250	218*	10
15	icmp	192.168.0.250	10.100.100.118	1
15	icmp	192.168.0.250	168.95.4.211	1
15	icmp	192.168.0.250	140.116.45.160	1
15	icmp	192.168.0.250	61.64.107.46	1
15	icmp	192.168.0.250	210*	8
15	icmp	192.168.0.250	60.182.90.5	1
15	icmp	192.168.150.120	194.242.113.133	1
15	icmp	192.168.150.120	213.186.60.106	1
15	icmp	192.168.150.120	63.217.27*	4

Rule	Protocol	SA	DA	Times
15	icmp	192.168.150.120	84.252.142.67	1
15	icmp	192.168.150.120	80.190*	2
15	icmp	192.168.150.120	216.156.142.13	1
15	icmp	192.168.150.120	207.226.112.22	1
15	icmp	192.168.150.120	205.177.3.24	1
15	icmp	192.168.190.206	203.68.243.10	1
15	tcp	192.168.140.59	222.62.182.224	1
21	tcp	192.168.10.8	210.241.22.19	1
23	tcp	210.241.0.243	210.69.91.69	1
24	tcp	192.168.10.8	210.241.22*	67
24	tcp	192.168.10.8	61.67.71*	17
27	tcp	172.10*	192.168.0.1	11
27	tcp	172.10.51.6	210.69.91.193	1
27	tcp	59*	210.69.91.193	7
27	tcp	203.90.122.165	210.69.91.193	1
27	tcp	210.75.1.85	210.69.91.193	1
27	tcp	218*	210.69.91.193	3
27	tcp	220*	210.69.91.193	3
27	tcp	221.191.250.132	210.69.91.193	1
27	tcp	222*	210.69.91.193	2
27	tcp	61*	210.69.91.193	8
27	tcp	80.88.144.137	210.69.91.193	1
28	tcp	192.168.10.3	192.168.1.30	1
29	tcp	172.10*	192.168.0.22	1181
29	tcp	192.168.130*	192.168.0.22	322
29	tcp	140.122.65.158	210.69.91.64	1
29	tcp	147.208.15.164	210.69.91.64	1
29	tcp	168.95.4*	210.69.91.64	10
29	tcp	192*	210.69.91.64	2
29	tcp	193.219.215.66	210.69.91.64	1
29	tcp	202*	210.69.91.64	45

Rule	Protocol	SA	DA	Times
29	tcp	203*	210.69.91.64	4
29	tcp	206.190.48.245	210.69.91.64	1
29	tcp	210*	210.69.91.64	16
29	tcp	211*	210.69.91.64	9
29	tcp	218*	210.69.91.64	5
29	tcp	219.81.164.195	210.69.91.64	1
29	tcp	220*	210.69.91.64	24
29	tcp	222.172.147.68	210.69.91.64	1
29	tcp	61*	210.69.91.64	13
29	tcp	65*	210.69.91.64	4
29	tcp	68.44.63.246	210.69.91.64	1
29	tcp	83.156.81.20	210.69.91.64	1
32	tcp	163.29.102.253	210.69.91.209	1
32	tcp	203.203.165.73	210.69.91.209	1
32	tcp	210*	210.69.91.209	9
32	tcp	211*	210.69.91.209	101
32	tcp	218*	210.69.91.209	630
32	tcp	219.81.224.77	210.69.91.209	1
32	tcp	220*	210.69.91.209	606
32	tcp	222*	210.69.91.209	8
32	tcp	59*	210.69.91.209	8
32	tcp	61*	210.69.91.209	50
32	tcp	68.142.251*	210.69.91.209	2
33	tcp	140*	210.69.91*	62
33	tcp	148.244.150.58	210.69.91.66	1
33	tcp	163.29.102.253	210.69.91.207	1
33	tcp	172.10*	192.168*	177
33	tcp	172.10.35.21	210.69.91.66	1
33	tcp	202*	210.69.91.66	4
33	tcp	203*	210.69.91*	110
33	tcp	207.46.98.66	210.69.91.66	1

Rule	Protocol	SA	DA	Times
33	tcp	210*	210.69.91*	293
33	tcp	211*	210.69.91*	783
33	tcp	218*	210.69.91*	3111
33	tcp	219*	210.69.91*	101
33	tcp	220*	210.69.91*	1464
33	tcp	221.169.108.124	210.69.91.207	1
33	tcp	222*	210.69.91*	36
33	tcp	59*	210.69.91*	252
33	tcp	60.248.84.10	210.69.91*	154
33	tcp	61*	210.69.91*	1585
33	tcp	65.54.188.66	210.69.91.66	1
33	tcp	66*	210.69.91*	58
33	tcp	68.142*	210.69.91.66	10
44	udp	192.168.130.21	192.168.140*	24
45	tcp	192.168.130*	203*	72
45	tcp	192.168.130*	210*	15
45	tcp	192.168.130*	202*	140
45	tcp	192.168.130.1	61*	40
45	tcp	192.168.130.1	205.161.7.31	1
45	tcp	192.168.130*	64*	4
45	tcp	192.168.130*	207*	5
45	tcp	192.168.130.1	66.218.72.54	1
45	tcp	192.168.130*	216*	12
45	tcp	192.168.130.29	211.78.38.199	1
45	tcp	192.168.130.29	65*	38
45	tcp	192.168.130.29	218.5.72.27	1
45	tcp	192.168.130.28	172.20.0.65	1
45	udp	192.168.130*	172.20.0*	18
47	tcp	172.10*	202*	499
47	tcp	172.10*	203*	631
47	tcp	172.10*	193.108.154*	12

Rule	Protocol	SA	DA	Times
47	tcp	172.10*	209.73.169.236	32
47	tcp	172.10*	61*	493
47	tcp	172.10*	205.161.7*	13
47	tcp	172.10*	211*	428
47	tcp	172.10*	66*	17
47	tcp	172.10*	210*	1825
47	tcp	172.10*	192*	306
47	tcp	172.10*	220*	53
47	tcp	172.10.50.2	72.9.241.122	1
47	tcp	172.10*	70*	13
47	tcp	172.10.50.2	38.114*	7
47	tcp	172.10*	216*	177
47	tcp	172.10*	64*	44
47	tcp	172.10*	67*	18
47	tcp	172.10.50.2	69.41.242.154	1
47	tcp	172.10*	65*	8
47	tcp	172.10.50.61	217.148.176.8	1
47	tcp	172.10*	207*	85
47	tcp	172.10.53.13	219.153.28.10	1
47	tcp	172.10.53.22	139.175.66.62	1
47	tcp	172.10*	140*	23
47	tcp	172.10*	208.172*	6
47	tcp	172.10.71*	163*	5
47	tcp	172.10.71.17	59.120.97.32	1
47	tcp	172.10.71.21	218.32.192.8	1
47	tcp	172.10.71.21	81.216.110.47	1
47	tcp	172.10.71.21	60.248.37.85	1
47	tcp	172.10.73.89	131.107.113.76	1
47	tcp	172.10.73.89	68.142.197.57	1
47	tcp	172.10*	168.95.4*	4
50	udp	192.168.254.254	210.69.91.254	1

Rule	Protocol	SA	DA	Times
52	tcp	61.30.139.145	210.69.91.70	1
60	tcp	172.10.35.11	192.168.0*	20
60	tcp	210*	210.69.91*	384
60	tcp	211*	210.69.91*	162
60	tcp	218*	210.69.91*	1142
60	tcp	219*	210.69.91*	351
60	tcp	221*	210.69.91*	106
60	tcp	59*	210.69.91*	283
60	tcp	60*	210.69.91*	20
60	tcp	61*	210.69.91*	748
60	udp	220*	210.69.91*	33
60	udp	221*	210.69.91*	2
60	udp	61*	210.69.91*	37
60	udp	70.85.177.186	210.69.91*	4
60	udp	213.78.111.100	210.69.91.10	1
60	tcp	213*	210.69.91*	5
60	tcp	194.54.222.113	210.69.91.6	1
60	udp	194.54.222.113	210.69.91.6	1
60	tcp	172.10*	203.69.40*	52
60	tcp	80.18.163.179	210.69.91*	59
60	tcp	81*	210.69.91*	2
60	udp	172.10.62.12	64.4.25*	17
60	udp	218*	210.69.91*	6
60	tcp	220*	210.69.91*	825
60	udp	59*	210.69.91.38	2
60	tcp	209.67.220.58	210.69.91.7	1
60	udp	172.10*	61.218.10.100	39
60	udp	64.4.12.201	210.69.91*	2
60	tcp	134.208.44*	210.69.91.6	2
60	tcp	140*	210.69.91*	98
60	tcp	163*	210.69.91*	21

Rule	Protocol	SA	DA	Times
60	tcp	202*	210.69.91*	164
60	tcp	203*	210.69.91*	305
60	tcp	222*	210.69.91*	202
60	tcp	24*	210.69.91*	23
60	tcp	58*	210.69.91*	10
60	tcp	68*	210.69.91*	3
60	tcp	71.104.1.223	210.69.91.6	1
60	tcp	69*	210.69.91*	4
60	udp	222.157.175.249	210.69.91.6	1
60	tcp	151*	210.69.91.7	11
60	tcp	192.192*	210.69.91*	9
60	tcp	195.252.72.14	210.69.91.7	1
60	tcp	62*	210.69.91*	2
60	tcp	66.25.126.86	210.69.91.7	1
60	tcp	82*	210.69.91.7	2
60	tcp	83.154.94.55	210.69.91.6	1
60	udp	83.154.94.55	210.69.91.6	1
60	udp	82.49.164.215	210.69.91.7	1
60	tcp	168.95*	210.69.91*	45
60	udp	203.94.238.230	210.69.91.3	1
60	tcp	201.226.116.28	210.69.91.64	1
60	tcp	172.10.71.21	61.31.238.28	1
60	udp	172.10*	207.46.130.100	9
60	udp	172.10.43.31	192*	2
60	udp	172.10.43.31	129.6.15.28	1
60	udp	172.10.43.31	132.163.4.102	1
60	udp	172.10.43.31	131.107.1.10	1
60	udp	172.10.43.31	128.2.181*	2
60	udp	172.10.43.31	69.25.27.170	1
60	tcp	147.208.15.164	210.69.91.90	1
60	tcp	193.219.215.66	210.69.91.90	1

Rule	Protocol	SA	DA	Times
60	tcp	200.149.220.182	210.69.91.90	1
60	tcp	206.190.48.245	210.69.91.90	1
60	tcp	212.242.110.56	210.69.91.90	1
60	tcp	65*	210.69.91.90	4
60	tcp	84.150.187.4	210.69.91.90	1
60	icmp	137.132.162.196	210.69.91.61	1
60	icmp	143.238.45.181	210.69.91.34	1
60	icmp	151*	210.69.91*	2
60	icmp	155*	210.69.91*	2
60	icmp	165.24.15.251	210.69.91.90	1
60	icmp	172.10*	192.168.0.22	2
60	41	172.10.62.25	131.107.33.60	1
60	41	172.10.62.25	192.88.99.1	1
60	icmp	172.10.70.24	202.43.195.52	1
60	icmp	193.77.152.162	210.69.91.34	1
60	icmp	194.225.77.5	210.69.91.23	1
60	icmp	195*	210.69.91*	2
60	icmp	196.33.198.63	210.69.91.2	1
60	icmp	200.177.101.130	210.69.91.76	1
60	icmp	202*	210.69.91*	4
60	icmp	203.173.48.236	210.69.91.7	1
60	icmp	204.196.142.2	210.69.91.22	1
60	icmp	206.174.67.165	210.69.91.10	1
60	icmp	207*	210.69.91*	2
60	icmp	209.11.89.34	210.69.91.20	1
60	icmp	210*	210.69.91*	3
60	icmp	211*	210.69.91*	2
60	icmp	216.203.33.134	210.69.91.13	1
60	icmp	218*	210.69.91*	11
60	icmp	219*	210.69.91*	4
60	icmp	220.201.133.235	210.69.91.72	1

Rule	Protocol	SA	DA	Times
60	icmp	221.232.138.131	210.69.91.8	1
60	icmp	24*	210.69.91*	4
60	icmp	59.58.194.185	210.69.91.28	1
60	icmp	60*	210.69.91*	3
60	icmp	61*	210.69.91*	27
60	icmp	63.116.151.124	210.69.91.72	1
60	icmp	64*	210.69.91*	2
60	icmp	66*	210.69.91*	2
60	icmp	67*	210.69.91*	3
60	icmp	68*	210.69.91*	7
60	icmp	69*	210.69.91*	3
60	icmp	70*	210.69.91*	4
60	icmp	71*	210.69.91*	4
60	icmp	80.63.59.10	210.69.91.28	1
60	icmp	81.130.208.246	210.69.91.27	1
60	icmp	82.40.113.143	210.69.91.8	1

附錄 2：

```
// aprioriProcess.cpp
// apply aprioriProcess.cpp to intrusion detection system
#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;


#include <fstream>
#include <string>

using std::ifstream;
using std::ofstream;
using std::string;

#include <iomanip>

#include <cstdlib> // exit prototype

using std::setprecision;

#include <cstdlib> // exit prototype
#include "tree.h" //Tree calss definition



int main()
{
```

```
    string Number, Date, Time, Interface, Action, Service, SourceIP, DestinationIP,
DestinationPort,    Protocol, Rule;
    string SourcePort;
    int Counter;
    Tree< string > stringTree; // create Tree of string values



  // ifstream constructor opens the file
  ifstream inLogFile( "C:\\c++\\twoStage\\0807\\0807.txt", ios::in );

  // exit program if ifstream could not open file
  if ( !inLogFile )
  {
      cerr << "File could not be opened" << endl;
      exit( 1 );
  } //end if

  // ofstream constructor opens the file
  ofstream outLogFile( "C:\\c++\\twoStage\\0807\\0807Output.txt", ios::out );

  // exit program if ifstream could not open file
  if ( !outLogFile ) {
      cerr << "File could not be opened" << endl;
      exit( 1 );
  } // end if



  while ( inLogFile >> Rule >> SourceIP >> DestinationIP >> Protocol >>
DestinationPort >> Action)
  {
        cout << Rule << "    " << SourceIP << "    " << DestinationIP << "    " <<
```

```
Protocol << " " << DestinationPort << " " << Action << "    " << endl;

            Counter = 1;
            stringTree.insertNode( Rule, SourceIP, DestinationIP, Protocol, SourcePort,
DestinationPort, Action, Counter); //Counter = 0

     }//end while

   cout << "\nPreorder traversal\n";
   stringTree.preOrderTraversal(outLogFile);



   inLogFile.clear();    //reset eof for next input
   inLogFile.seekg(0); //move to beginning of file


  return 0; // ifstream destructor closes the file
}
```

```
//Fig. 17.18: tree.h
//Template Tree class definition.
#ifndef TREE_H
#define TREE_H

#include <iostream>
#include <sstream>
#include <string>
#include <stdlib.h>
#include <stdio.h>
#include <fstream>


using std::endl;
using std::ifstream;
using std::ofstream;

#include <new>
#include "treenode.h"

int dupTime = 0;



template<class NODETYPE>
class Tree {

public:
      Tree();
      void insertNode( const NODETYPE & , const NODETYPE &, const NODETYPE
&, const NODETYPE &, const NODETYPE &, const NODETYPE &, const
NODETYPE &, int &);
      void preOrderTraversal( ofstream &); //const;
```

```
private:
      TreeNode< NODETYPE > *rootPtr;

      // utility functions
      void insertNodeHelper(
            TreeNode <NODETYPE> **, const NODETYPE &, const NODETYPE &,
const NODETYPE &, const NODETYPE &, const NODETYPE &, const NODETYPE
&, const NODETYPE &, int &);
      void checkIP( const NODETYPE &, const NODETYPE &, NODETYPE &);
      void preOrderHelper( TreeNode<NODETYPE> *, ofstream &); //const;

}; //end class Tree

// constructor
template< class NODETYPE >
Tree< NODETYPE >::Tree()
{
   rootPtr = 0;
  } // end Tree constructor

// insert node in Tree
template< class NODETYPE >
void Tree< NODETYPE >::insertNode( const NODETYPE &tRule, const NODETYPE
&SourceIP, const NODETYPE &DestinationIP, const NODETYPE &Protocol, const
NODETYPE &SourcePort, const NODETYPE &DestinationPort, const NODETYPE
&Action, int &Counter)
{
   insertNodeHelper( &rootPtr,    tRule, SourceIP, DestinationIP, Protocol, SourcePort,
DestinationPort, Action, Counter);
}// end function insertNode

// utility function called by insertNode; receives a pointer
```

```
// to a pointer so that the function can modify pointer's value
template< class NODETYPE >
void Tree< NODETYPE >::insertNodeHelper(
      TreeNode< NODETYPE > **ptr, const NODETYPE &tRule, const NODETYPE
&SourceIP, const NODETYPE &DestinationIP, const NODETYPE &Protocol, const
NODETYPE &SourcePort, const NODETYPE &DestinationPort, const NODETYPE
&Action, int &Counter)
{
   string tempKey1, tempKey2, tempSourceIP, tempDestinationIP, temp;
   string sDupIP, dDupIP;

   tempKey1 = tRule + Protocol + Action + SourceIP + DestinationIP;

   if (*ptr == 0)
        *ptr = new TreeNode< NODETYPE >( tRule, SourceIP, DestinationIP, Protocol,
SourcePort, DestinationPort, Action, Counter); //Counter = 0
   else //subtree is not empty
   {
        tempKey2 = ( *ptr ) ->Rule + ( *ptr ) ->Protocol + ( *ptr ) ->Action + ( *ptr )
->SourceIP + ( *ptr ) ->DestinationIP;
        //data to intsert is less than data in current node
        if ( tempKey1 < tempKey2 ){
             temp = tRule + Protocol + Action;
            if (tempKey2.find(temp) != 0)
                insertNodeHelper(&((*ptr)->rightPtr), tRule, SourceIP, DestinationIP,
Protocol, SourcePort, DestinationPort, Action, Counter);
            else {
                 checkIP(SourceIP, (*ptr)->SourceIP, sDupIP);
                 checkIP(DestinationIP, (*ptr)->DestinationIP, dDupIP);
                 if ((sDupIP != "") && (dDupIP != "")){
                     if (((*ptr)->SourceIP) != sDupIP)
                          (*ptr)->SourceIP = sDupIP + "*";
                     if (((*ptr)->DestinationIP) != dDupIP)
```

```
                    (*ptr)->DestinationIP = dDupIP + "*";
                (*ptr)->Counter ++;
                cout << "(*ptr)->SourceIP=    " << (*ptr)->SourceIP <<
"(*ptr)->DestinationIP=    " << (*ptr)->DestinationIP << endl;
            } //if ((sDupIP != "") && (dDupIP != ""))
            else
                insertNodeHelper(&((*ptr)->rightPtr), tRule, SourceIP, DestinationIP,
Protocol, SourcePort, DestinationPort, Action, Counter);
        }//else
       } //if ( tempKey1 < tempKey2 )
            else
      //data to insert is greater than data in current node
      if ( tempKey1 > tempKey2){
          temp = tRule + Protocol + Action;
        if (tempKey2.find(temp) != 0)
            insertNodeHelper(&((*ptr)->rightPtr), tRule, SourceIP, DestinationIP,
Protocol, SourcePort, DestinationPort, Action, Counter);
        else {
            checkIP(SourceIP, (*ptr)->SourceIP, sDupIP);
            checkIP(DestinationIP, (*ptr)->DestinationIP, dDupIP);
            if ((sDupIP != "") && (dDupIP != "")){
                if (((*ptr)->SourceIP) != sDupIP)
                    (*ptr)->SourceIP = sDupIP + "*";
                if (((*ptr)->DestinationIP) != dDupIP)
                    (*ptr)->DestinationIP = dDupIP + "*";
                (*ptr)->Counter ++;
                cout << "(*ptr)->SourceIP=    " << (*ptr)->SourceIP <<
"(*ptr)->DestinationIP=    " << (*ptr)->DestinationIP << endl;
            } //if ((sDupIP != "") && (dDupIP != ""))
            else
                insertNodeHelper(&((*ptr)->rightPtr), tRule, SourceIP, DestinationIP,
Protocol, SourcePort, DestinationPort, Action, Counter);
```

```
            } //else
        } //if ( tempKey1 > tempKey2)
                else //duplicate data value ignored
                cout << tempKey1 << " dupTime= " << dupTime ++<< endl;



    }
} // end function insertNodeHelper

// begin checkIP
template< class NODETYPE >
void Tree< NODETYPE >::checkIP( const NODETYPE &IP, const NODETYPE &ptrIP,
NODETYPE &DupIp)
{
    string tempIP, IP1, IP2, IP3, IP4;
    //string DupIP;
    int Location;

    tempIP = IP;
    IP4 = tempIP;
    Location = tempIP.rfind(".");
    IP3 = tempIP.substr(0, Location);
    Location = IP3.rfind(".");
    IP2 = IP3.substr(0, Location);
    Location = IP2.rfind(".");
    IP1 = IP2.substr(0, Location);


    if (ptrIP == IP4)
        DupIp = IP4;
    else if (ptrIP.find(IP3) == 0)
        DupIp = IP3;
```

```
   else if (ptrIP.find(IP2) == 0)
         DupIp = IP2;
   else if (ptrIP.find(IP1) == 0)
         DupIp = IP1;

} // end function preOrderTraversal


// begin preorder traversal of Tree
template< class NODETYPE >
void Tree< NODETYPE >::preOrderTraversal( ofstream &outLogFile) //const//, const
{
    preOrderHelper( rootPtr, outLogFile);

} // end function preOrderTraversal

//utility function to perform preorder traversal of Tree
template< class NODETYPE >
void Tree< NODETYPE >::preOrderHelper(
      TreeNode< NODETYPE > *ptr, ofstream &outLogFile )
{


      if ( ptr != 0 ) {
          preOrderHelper( ptr->leftPtr, outLogFile ); //go to left subtree
            outLogFile    << ptr-> Rule << "    " << ptr-> Protocol << "    " << ptr->
SourceIP << "    " << ptr-> DestinationIP <<    "    " << ptr-> Counter << endl;

          preOrderHelper( ptr->rightPtr, outLogFile); // go to right subtree
      }// end if
} // end function preOrderHelper
```

```
#endif
```

```
//Fig 17.17: treenode.h
//Template TreeNode class definition
#ifndef TREENODE_H
#define TREENODE_H

//forward declaration of class Tree
template< class NODETYPE> class Tree;

template< class NODETYPE >
class TreeNode {
      friend class Tree< NODETYPE >;

public:

      //constructor
      TreeNode( const NODETYPE &r, const NODETYPE &s, const NODETYPE &d,
const NODETYPE &p, const NODETYPE &sp, const NODETYPE &dp, const
NODETYPE &a, int &c)
            : leftPtr( 0 ),
               Rule ( r ),
               SourceIP ( s ),
               DestinationIP ( d ),
               Protocol ( p ),
               SourcePort ( sp ),
               DestinationPort ( dp ),
               Action ( a ),
               Counter ( c ),
               rightPtr ( 0 )
      {
         // empty body

      } // end TreeNode Constructor
```

```
    // return copy of node's data
    NODETYPE getData()    const
    {
        return Rule;
    } // end getdata function

private:
    TreeNode< NODETYPE > *leftPtr; // pointer to left subtree
    NODETYPE Rule;
    NODETYPE SourceIP;
    NODETYPE DestinationIP;
    NODETYPE Protocol;
    NODETYPE SourcePort;
    NODETYPE DestinationPort;
    NODETYPE Action;
    int Counter;
    TreeNode< NODETYPE > *rightPtr; // pointer to left subtree


}; // end class TreeNode

#endif
```