樹德科技大學
資訊工程系碩士班

碩士論文

RST 入侵偵測預處理機制

研究生：洪弘洲

指導教授：曾昱國

中華民國　九十七　年　六　月

RST 入侵偵測預處理機制

Preprocess Mechanism of Intrusion Detection Based on RST

研究生：洪弘洲
指導教授：曾昱國

樹德科技大學
資訊工程系碩士班
碩士論文

A Thesis
Submitted to the Graduate Division in
Partial Fulfillment of the Requirements for the Degree of Master of Science in
Department of Computer Science and Information Engineering

Shu-Te University
Kaohsiung County, Taiwan
Republic of China
July 1, 2008

中華民國九十七年七月

RST 入侵偵測預處理機制

學生：洪弘洲　　指導教授：曾昱國 博士

樹德科技大學資訊工程系碩士班

摘要

目前對於入侵偵測系統(Intrusion Detection System ; IDS) 的主要研究方向是針對提高偵測準確率和降低系統負擔。由於入侵偵測系統漏過了任何攻擊都可能對電腦產生極大威脅，所以提高偵測準確率是許多研究者的研究目標，但是入侵偵測需要處理的資料量十分龐大，因此往往為了提高偵測準確率，卻使得效能大幅降低，所以便需使用資料預處理相關技術來提升入侵偵測系統效能。

目前資料預處理中許多研究均以去除品質不良資料為主要研究對象，但這些研究所提出的預處理方法，不是無科學根據地移除資料，就是移除的資料對於提升入侵偵測系統的偵測準確性上沒有幫助，甚至還降低了偵測準確率。所以我們提出以約略集理論(Rough Set Theory)的方法來分析並找出蒐集的資料中對於提升攻擊偵測率沒有幫助的特徵屬性並移除它們，而經由縱向的屬性縮減後，將可大幅減少所需分析的資料量，使入侵偵測系統達到最佳的效能， 經由本論文的實驗證明我們的方法確實能減少一半左右屬性的資料量卻不會降低偵測準確率。

我們也和其他使用 KDD-Cup1999 資料集的相關研究進行比較，分析之間優

劣，證明我們所使用的方法並不亞於其他相關研究，我們有效的刪除不必要的屬性還能保持原有較高的準確率及覆蓋率。最後本研究也將預處理結果分別套至入侵偵測系統中常見的不同分類方法來進行實驗，結果證明本研究成果有助於入侵偵測系統在資料量減少下仍可維持良好的準確率及覆蓋率。

關鍵詞：入侵偵測、資料預處理、約略集理論、屬性縮減

Preprocess Mechanism of Intrusion Detection Based on RST

Student：Hong-Zhou Hong　　Advisors：Dr. Yu-Kuo Tseng

Department of Computer Science and Information Engineering
Shu-Te University

Abstract

At present, the main research direction for Intrusion Detection System (IDS) is to enhance the detecting accurate rate, reduce the false positive rate, and improve the system performance. The amount of data, which intrusion detection needs to process, is usually extremely huge, so it is an important issue for IDS to improve its performance through reducing the amount of data. Under high overload, though, IDS might omit some attack attempts that could cause potential and serious threats to user's computers. Thus, the detecting rate will decrease as well.

As to reducing the amount of data, many preprocessing schemes are also proposed to enhance the IDS performance. However, without any reasonable and scientific explanation, most of them just remove some data from the data set required to be analyzed by IDS. Furthermore, removing those data incorrectly could lower the detecting rate. Therefore, we propose a scientific IDS preprocessing scheme based on Rough Set Theory to discover and remove around a half useless conditional attributes. Pruning helpless attribute subset will refine data set and boost the IDS performance.

We have compared our research with others which also use KDD-Cup1999 data set, and the experiment results show that the proposed scheme's accuracy rate and coverage rate is not worse than others, after removing. The preprocessed and reducted KDD-Cup data set also is input into three common classification medthods used in the

kernal of Intrusion Detection System. There is Decomposition Tree, Neural Network, K-th nearest neighbor, and every detecting rate between the KDD-Cup 1999 data set is the same. Therefore, the proposed scheme will boost the performance of IDS without effecting the IDS detecting rate.

Keyword : Intrusion Detection、Data preprocessing、Rough Set Theory、Attribute Reduct

誌 謝

感謝曾昱國教授兩年來的指導，從我進入樹德科技大學碩士班尚無明確目標一點一滴的帶入我的研究領域，最後完成我的論文研究，也在我的研究主題給予我十分多的建議與指教，最後也針對我論文撰寫部份給予我許多的寶貴意見，使我的論文撰寫部分更容易理解，因此十分感謝曾昱國教授給予我的種種幫助，使我能夠完成我的研究。感謝班上的每一位同學陪我走過這兩年苦悶的日子，雖然在研究時遇到許多問題，但也感謝余適印同學幫我解決研究上的一些問題以及廖堉崴同學和夏傑興同學給我論文上面的一些建議，也感謝余適印同學、廖堉崴同學、張立志同學、鄭永隆同學等在我煩悶時陪我玩許多的小遊戲，度過許多無聊又煩悶的時光，讓我再有動力繼續進行我的研究。最後也感謝我的家人，父母給我的支持，使我有更有動力完成我的研究。

洪弘洲 謹誌

2008 年 7 月

目錄

圖目錄

表目錄

一、 緒論

1.1 研究背景

隨著時代的進步，電腦的發展更是日新月異，網路頻寬以及流量也日益龐大，而在資訊的快速發展下，雖然使我們的生活方便很多，但也出現了許多問題，如網路世界中出現越來越多攻擊手法及攻擊軟體，而在攻擊軟體越來越容易得到且十分容易操作的環境下，也被越來越多居心不良的駭客所利用來進行攻擊和入侵，因此許多一般使用者所使用的電腦也就十分不安全，隨時可能被居心不良的駭客攻擊，導致電腦資料失竊或是電腦癱瘓，可能使得個人私密資料被竊取，更有可能機密資料被盜取或是遺失，使得受害者平白無故的遭受到這些損失。

因此針對這些問題的發生，越來越多的入侵偵測系統(Intrusion Detection System)的研究開始出現，這些研究開始朝著增加偵測能力來偵察新型攻擊，而系統為了處理因為攻擊而產生的龐大網路流量或系統稽核紀錄等資料，使得系統負擔越來越沉重，因而可能導致入侵偵測系統效能變差，最後甚至可能使得入侵偵測系統因為無法及時處理龐大流量而放過許多攻擊封包，或是將許多正常封包阻擋在系統之外，因此如何增加入侵偵測系統的效能，而不使系統偵測準確率降低，便是許多研究探討的重要議題。

目前入侵偵測系統主要有兩種方式來判斷入侵攻擊，第一種為特徵比對(Misuse Detection)，判斷方式是依照若與已知的攻擊特徵資料相同或類似時，則判斷為入侵行為；第二種為異常偵測(Anomaly Detection)，判斷方式是訊息的行為如果超出先前訓練得到的正常統計範圍之外，則判斷為入侵行

為[1]。而本研究針對入侵訊務及系統紀錄等龐大資料來進行預處理分析，找出能協助入侵偵測系統判斷的的所需最小資料量以提升入侵偵測的效能。

1.2 研究動機與目的

基於目前網路上的環境，網路的發展十分的快速，網路流量也日益龐大，所以入侵偵測系統的負荷便也跟著相對提升，因此我們的研究便著重於去除對於判斷正常與異常封包沒有幫助的訊息，因為在龐大的資料訊息中一定有許多資訊對於入侵偵測系統探勘攻擊訊務是沒有幫助的，若能夠找出這些資訊並移除它們，便能夠使入侵偵測系統只針對有助於分類正常與攻擊類型的特徵資訊去進行資料探勘的工作，進而減少入侵偵測系統的工作量，使入侵偵測系統的效能獲得提升。

而為了使入侵偵測系統用來進行探勘的資料量降低，我們針對資料進行預先處理。資料預處理的主要目的就是解決資料品質不良的問題，而未經處理的資料可能存在著許多問題，例如資料不完整、資料有雜訊、資料不一致等問題，導致資料探勘得不到最佳效果，所以資料預處理是一項十分重要的議題。而資料預處理主要包含資料整合(data integration)、資料清理(data cleaning)、資料轉換(data transformation)、資料精簡(data reduction)[2]。本研究主要針對資料精簡的部份進行預處理研究，處理方法從資料集中濾掉一些無關的或是無幫助的訊息，因為資料探勘的過程，隨著資料量變多，需要花費更多的處理時間，因此挑選少量且具代表性的資料將大幅縮減資料探勘所需的時間。而資料精簡主要有兩個重點來幫助我們的研究思考，一為代表性的資料選取，如何評核原始資料，並從中挑選出重要、有助於知識獲得的資料；二為資料精簡的方法，在資料精簡的過程中，是否存在有效率的方法來進行。因此在本研究中我們使用約略集理論建立訊息系統去分析資料，挑選出不需要的資料進行刪減，並找出最精簡的屬性，進而幫助入侵偵測系統提升效能。

本研究主要是針對入侵偵測的資料來進行預處理，而 KDD(Knowledge Discovery and Data)-Cup[3]的資料集也是最常被使用來進行資料探勘的研究，因此我們使用 KDD-Cup 中專門為了電腦網路入侵偵測所製作的 KDD-Cup1999 資料集來當我們的實驗資料集。此資料集中包含大量正常流量及攻擊流量，我們希望經過資料預處理後，刪減掉一些對偵測沒幫助的屬性，進而降低大量資料，而資料量降低便能夠減少入侵偵測系統的處理量，使得系統得以快速判斷出攻擊封包，因此隨著資料降低，系統的處理便會加快，使得效能提升，更適應現在及未來流量龐大的網路環境。

1.3 論文架構

本篇論文其他章節架構如下，第二章文獻探討，介紹本篇所使用的資料集及使用 KDD-Cup 1999 資料集的相關研究，還有目前研究預處理以及屬性選取的相關論文。第三章介紹我們所提出的二階段屬性縮減的資料預處理架構及機制。第四章為實驗結果分析與探討，呈現我們使用 KDD-Cup 1999 資料集實驗後所得到的實驗結果及與其他各種分類器進行比較的結果。第五章為結論與未來工作，為我們研究作一個總結，並敘述未來我們仍須對哪些問題繼續探討與解決，使本研究能夠更加完整。

二、 文獻探討

我們將在 2.1 節針對 KDD-Cup 1999 資料集做一些介紹;接著 2.2 節我們會介紹幾篇使用 KDD-Cup 1999 做入侵偵測的研究，在 2.3 節介紹目前預處理的相關研究，在 2.4 節介紹本研究所使用的約略集理論。

2.1 KDD-Cup 1999 資料集

KDD-cup[3]自 1997 年至 2006 年均提供許多類型的資料集,如 KDD-Cup 1998 資料集針對直接營銷利潤優化、KDD-Cup 2000 針對線上零售商的網站點閱率分析等等 KDD-Cup 資料集，而 KDD-Cup 1999 主要是針對網路入侵偵測的資料集，因此本研究也和其他研究一樣均以此資料集為研究標的。

0 tcp http SF 232 2654 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 16 17 0 0 0 0 1 0 0.12 206 242 1 0 0 0.04 0 0 0 0 normal.
0 tcp http SF 288 1046 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 3 255 1 0 0.33 0.03 0 0 0 0 normal.
0 tcp http SF 345 6055 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 8 13 0.12 0.08 0 0 1 0 0.31 8 255 1 0 0.12 0.08 0.12 0.01 0 0 normal.
0 tcp http SF 164 2112 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 8 10 0 0 0 0 1 0 0.2 34 255 1 0 0.03 0.03 0 0 0 0 normal.
0 tcp http SF 273 4471 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 24 24 0 0 0 0 1 0 0 136 255 1 0 0.01 0.04 0 0 0 0 normal.
0 tcp smtp SF 3446 321 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 2 3 0 0 0 0 1 0 0.67 114 183 0.91 0.03 0.01 0.01 0 0 0 0 normal.
0 tcp http SF 297 770 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 5 42 0 0 0 0 1 0 0.05 60 255 1 0 0.02 0.04 0 0 0 0.01 normal.
0 tcp http SF 207 376 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 3 20 0 0 0 0 1 0 0.35 29 255 1 0 0.03 0.04 0 0 0 0 normal.
0 tcp private S0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 129 3 1 1 0 0 0.02 0.08 0 255 3 0.01 0.08 0 0 1 1 0 0 neptune.
0 tcp http SF 287 1572 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 6 7 0 0 0 0 1 0 0.28 13 255 1 0 0.08 0.03 0 0 0 0 normal.
0 tcp private S0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 119 14 1 1 0 0 0.12 0.07 0 255 14 0.05 0.06 0 0 1 1 0 0 neptune.
0 tcp http SF 207 1918 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 13 13 0 0 0 0 1 0 0 255 255 1 0 0 0 0 0 0 0 normal.
0 tcp private S0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 231 16 1 1 0 0 0.07 0.06 0 255 16 0.06 0.09 0 0 1 1 0 0 neptune.
0 udp domain_u SF 43 76 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 90 90 0 0 0 0 1 0 0 255 255 1 0 0.01 0 0 0 0 0 normal.
0 tcp private S0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 131 6 1 1 0 0 0.05 0.06 0 255 14 0.05 0.07 0 0 1 1 0 0 neptune.
0 tcp http SF 270 9590 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 9 9 0 0 0 0 1 0 0 9 255 1 0 0.11 0.07 0 0 0 0 normal.
0 tcp http REJ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 1 2 1 0 1 1 0 0 1 1 normal.
0 tcp ftp_data SF 383 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 29 50 0.31 0.14 0.31 0.04 0 0 0 0 normal.
0 tcp http SF 248 1249 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 200 255 1 0 0 0.01 0 0 0 0 normal.
0 udp other SF 146 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 1 0 0 0 0 0.33 0.67 0 255 1 0 0.82 0.97 0 0 0 0 0 normal.

圖 1 ： KDD-Cup 1999 資料集範例

KDD-Cup 1999 資料集的產生是在一個模擬實驗的網路環境，進行為期 9 週的網路資料收集，其間除了蒐集正常的封包資料外，更模擬蒐集了多種不同攻擊行為的封包資料，所收集到的封包資料轉換成網路連線資料與系統存取資料，並且加入專家所建議的其他網路統計與系統資訊,資料集中共有 42 個與基本網路連線、網路統計和系統資訊相關的特徵屬性[3]，且對每一筆資料記錄給予正常或屬於哪種攻擊行為的標籤，資料集樣本如圖 1。

KDD Cup 1999 資料集屬性特徵數目共有 42 種，其中 41 種為條件屬性，1 種為決策屬性，如下圖 2，其中各屬性代表意義如表 1:

而攻擊行為主要可以分為 DoS、Probe、U2R、R2L 四類[3]:

- DoS :如 Ping of Death、SYN Flooding、smurf 等。

- Probe : 探測被掃描端的位址、可能的作業系統版本、提供網路服務及相關的埠號，接著可以利用提供服務軟體本身的弱點或是軟體的設定錯誤，得到未經認可的使用權限。

- U2R : 此攻擊是指合法的一般使用者或非法獲得的一般使用者權限經由緩衝區溢位等攻擊方式，得到超級使用者的權限。

- R2L : 遠端攻擊者利用如 netBIOS、NFS、finger 等服務，發現可利用的使用者帳號以及不適當的使用者設定，非法登入主機。此類攻擊如 U2R 一樣，除了網路封包技巧，如欺騙的應用外，另需配合系統的錯誤設定與軟體弱點的應用。

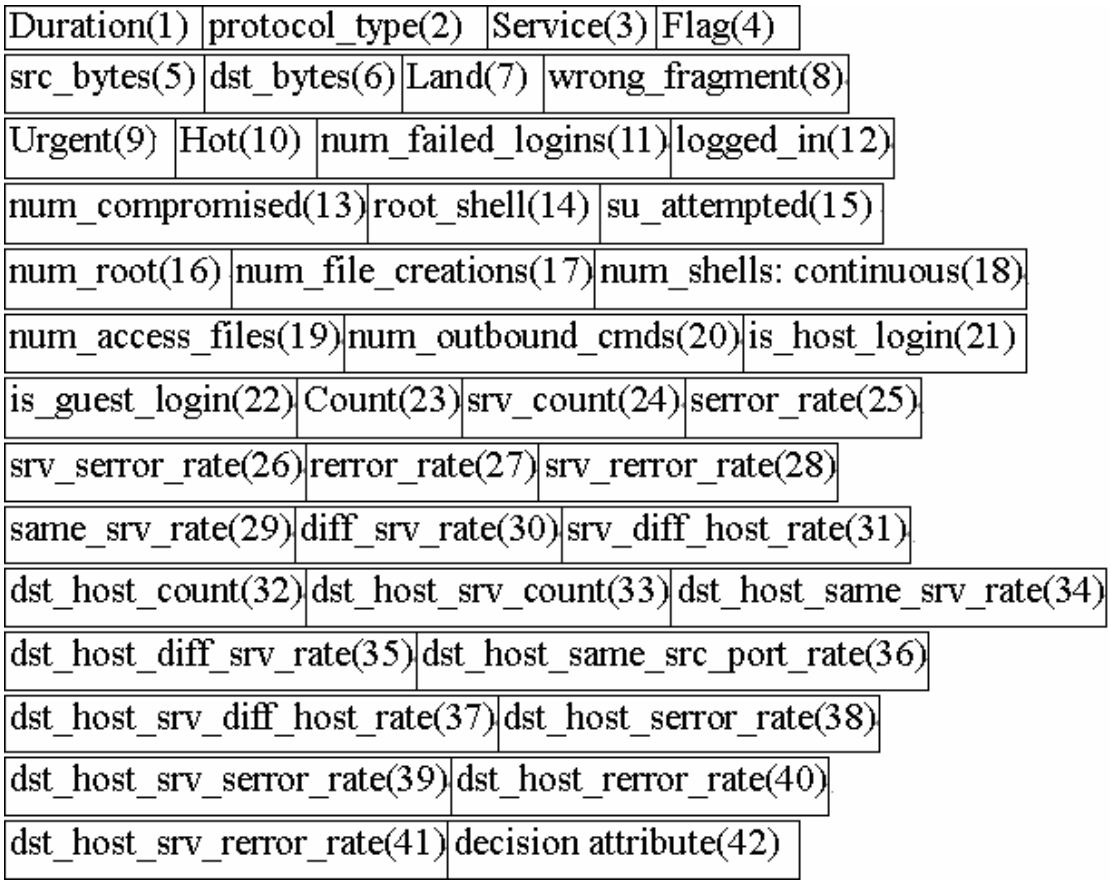

圖 2 ： KDD Cup 1999 資料集屬性特徵

表 1 ： KDD-Cup 1999 資料集條件屬性簡介

屬性名稱	屬性代表意義	屬性類型
Duration(1)	表示連線的時間長度	連續性
Protocol_type(2)	表示所屬的協定，如 tcp、udp 等等	符號型
Service(3)	網路服務名稱，目的端的網路服務，如 http、telnet 等等	符號型
Flag(4)	狀態，代表某連線正常或異常的狀態	符號型
Src_bytes(5)	傳送資料量，從來源端(source)傳輸到目的端的資料量	連續性

Dst_bytes(6)	接收資料量，從目的端(destination)傳輸到來源端的資料量	連續性
Land(7)	假如來源端和目的端都來自相同主機和通訊埠則設為 1，反之設為 0	符號型
Wrong_fragment(8)	錯誤區段數量，表示錯誤區段的數量	連續性
Urgent(9)	緊急封包數量，表示緊急封包的數量	連續性
Hot(10)	路由器數量，表示通過的路由器數量	連續性
Num_failed_logins(11)	登入失敗次數，表示企圖登入失敗的次數	連續性
Logged_in(12)	登入狀態，登入成功設為 1，登入失敗設為 0	符號型
Num_compromised(13)	危害的連線個數	連續性
Root_shell(14)	root 權限狀態，假如最高的權限被獲得則設為 1，反之設為 0	連續性
Su_attempted(15)	如果有 su root 的命令企圖則為 1，反之設為 0	連續性
Num_root(16)	嘗試轉換最高權限者(root)的次數	連續性
Num_file_creations(17)	創建檔案操作的數量	連續性
num_shells(18)	shell 的數量	連續性
num_access_files(19)	存取並操作檔案的次數	連續性
num_outbound_cmds(20)	在一個連結中訪問外部的命令個數	連續性
is_host_login(21)	如果是以主機(host)身分登入則設為 1，反之則設為 0	符號型
is_guest_login(22)	如果是以來賓(guest)身分登入則設為 1，反之則設為 0	符號型

count(23)	連線數量，過去 2 秒內連接到與目前連線相同服務的連線數量	連續性
srv_count(24)	服務數量，過去 2 秒內與目前連線所指服務一樣的連接個數	連續性
serror_rate(25)	SYN 錯誤率，過去 2 秒內，連接到與目前連線相同主機所產生 SYN 錯誤（亦即狀態 (flag) 為 S0）的連線比率	連續性
srv_serror_rate(26)	過去 2 秒內，連接到與目前連線相同主機的服務所產生 SYN 錯誤（亦即狀態的連線比率	連續性
rerror_rate(27)	REJ 錯誤率，過去 2 秒內，連接到與目前連線相同主機所產生 REJ 錯誤（狀態為 REJ）的連線比率	連續性
srv_rerror_rate(28)	過去 2 秒內，連接到與目前連線相同主機的服務所產生 REJ 錯誤（狀態為 REJ）的連線比率	連續性
same_srv_rate(29)	過去 2 秒內，連接到與目前連線相同主機並要求相同服務的連線比率	連續性
diff_srv_rate(30)	過去 2 秒內，連線到與目前連線相同主機但要求不同服務的連線比率	連續性
srv_diff_host_rate(31)	過去 2 秒內，連線到與目前連線不同主機且要求服務的連線比率	連續性
dst_host_count(32)	同屬性 23，不過為目的主機	連續性
dst_host_srv_count(33)	同屬性 24，不過為目的主機	連續性

dst_host_same_srv_rate(34)	同屬性 29，不過為目的主機	連續性
dst_host_diff_srv_rate(35)	同屬性 30，不過為目的主機	連續性
dst_host_same_src_port_rate(36)	過去 2 秒內，目的主機連線到與目前連線有相同服務及埠號的連線比率	連續性
dst_host_srv_diff_host_rate(37)	同屬性 31，不過為目的主機	連續性
dst_host_serror_rate(38)	同屬性 25，不過為目的主機	連續性
dst_host_srv_serror_rate(39)	同屬性 26，不過為目的主機	連續性
dst_host_rerror_rate(40)	同屬性 27，不過為目的主機	連續性
dst_host_srv_rerror_rate(41)	同屬性 28，不過為目的主機	連續性

2.2 KDD-Cup 1999 相關研究

目前入侵偵測的研究有許多，但主要的研究重點都是提升偵測準確率或是提升效能，而這些研究所使用的實驗資料集大多都是來自 KDD-Cup，因此我們將針對與我們研究使用相同 KDD-Cup 1999 資料集的研究進行介紹與探討

1. S.Peddabachigari 等人使用混合兩種技術，分別為決策樹(Decision Tree)和支援向量機(SVM)去偵測 DoS/DDoS 攻擊，希望藉由結合使用 ID3 演算法的決策樹以及使用多項式內核(Polynomial Kernel Function)的支援向量機這兩種分類器的分類技術來分別針對不同型態的攻擊資料進行分類，使系統達到更高偵測準確率，將所有類別區分為五個大類別，分別為 Normal、DoS、U2R、Probe、R2L，系統模型如圖 3，對於資料集分別使用決策樹(DT)來判斷 U2R 類型的攻擊、支援向量機(SVM)來判斷 DoS 類型的攻擊、並結合決策樹和支援向量機(HyBird DT-SVM)來判斷 Normal 的類型，最後結合 DT、SVM 及混合型的來判斷 Probe 和 R2L 的類型，最後使用 KDD-Cup1999 的資料集進行實驗，實驗證明在資料量小(6890 筆資料)的情況下的確實能夠使

U2L、Probe 及 R2L 準確率明顯的獲得提升[4]。

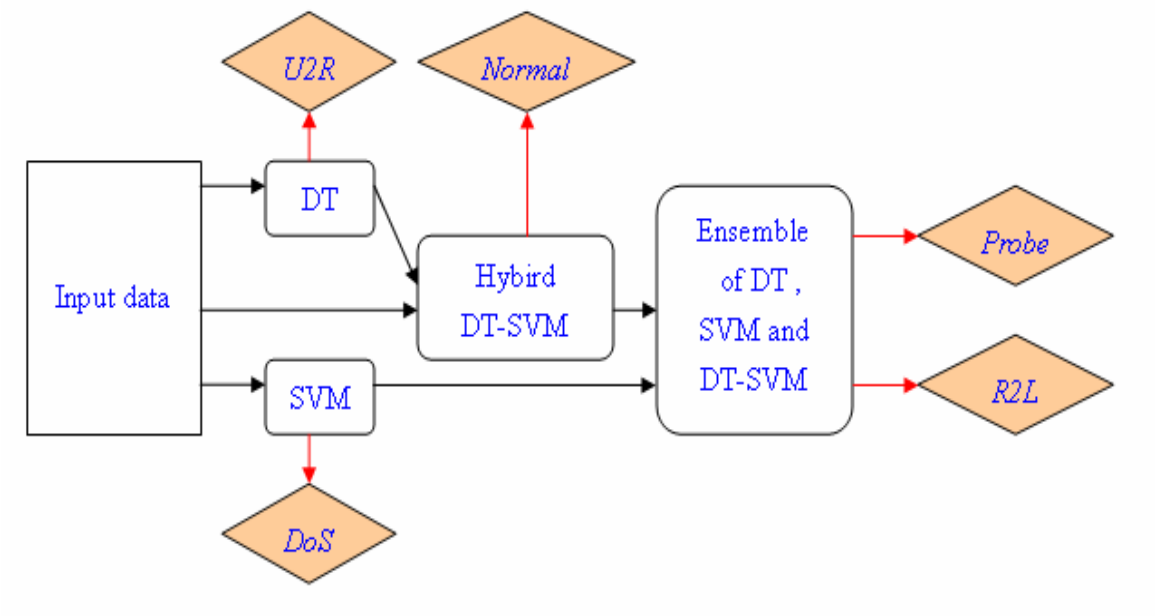

圖 3 ： 階層式智慧型入侵偵測系統

但是由於結合了兩種技術也使得系統的複雜度更高導致效能大幅下降，而且由於此篇研究所使用的訓練集及測試集資料分別為 5092 筆和 6890 筆，明顯實驗資料量小，所以由決策樹和支援向量機個別的效率看起來並不會太差，但是如果應用於真實環境下資料量龐大，便可能使得效率大幅降低，尤其是混合型的分類器，而此篇也沒有提到應用預處理的技術去解決問題，因此若能夠有效的配合資料預處理使資料量降低，減少系統的負荷便能夠達到偵測準確率的提升而不會降低效能，而如何能夠達到這個目的便能夠配合本研究的方法。

2. H. Hom 等人針對要辨別的攻擊定義一個分類演算法去對入侵資料進行分類，分類方法是使用特徵空間分類模型去辨識 DDoS 攻擊，詳細的方法介紹如下[5]:

對於任何兩個協方差矩陣A和B，定義一個變數$C = A - B$，藉由得到一個門檻值δ，改變C的值，接著定義一個新的矩陣$D_{A,B} = D(C,\delta)$，如下式:

$$\forall d_{i,j} \in D, \forall c_{i,j} \in C$$

$$d_{i,j} = \begin{cases} 1 \ \ if \mid c_{i,j} \mid \geq \delta \\ 0 \ \ if \mid c_{i,j} \mid < \delta \end{cases}$$

D和C是相同維度，我們使用δ當門檻值來評估兩個協方差矩陣之間顯著的差別，至於對於A和B的差異，假如差異大於δ值，則D的值將設定為 1，反之則為 0，換句話說就是對於任何配對的攻擊類型，假如他們在新的特徵空間的差異大於一個取得的門檻值，則在D裡對應的$d_{i,j}$將等於 1。

實驗使用 KDD-Cup 1999 資料集並利用 H. Hom 等人提出的分類演算法進行分類，並從資料集原本 41 種特徵屬性中挑選 32 種屬性去建構協方差特徵空間，文中提到不使用其他 9 個物理特徵不是屬於離散的就是符號的屬性特徵類型，不能夠使用不合理的資料預處理方法，由此可知，此研究只針對特徵空間的需求選擇了不屬於離散型和符號類型的屬性特徵，因此可能使用了過多的資料進行處理，若資料量龐大或攻擊類型繁多，則可能導致太多的資料需要經由此研究提出的分類演算法進行比較，因此導致系統的效能降低，若能夠針對資料進行妥善的預處理動作，找出對於分類演算法沒有幫助的多餘資料，則能夠在提升系統準確率的情況下確能夠不降低其效能。

3. S.Mukkamala針對DoS型攻擊使用支援向量機(SVM)來偵測，分別使用S. Mukkamala之前研究[6]提出的屬性數目來進行實驗，他提出20種第一重要的屬性和12種次等重要的屬性以及9個不重要的屬性，而他的實驗中分為兩部份，第一部份使用重要性的等級主要針對效能，分別使用了20個

第一重要{ duration、 service、src_bytes、dst_bytes、wrong_fragment、num_access_files、count、srv_count、serror_rate、srv_serror_rate、rerror_rate、srv_rerror_rate、dst_host_count、dst_host_srv_count、dst_host_diff_srv_rate、dst_host_same_src_port_rate 、 dst_host_serror_rate 、 dst_host_srv_serror_rate、dst_host_rerror_rate、dst_host_srv_rerror_rate}和加上12個次等重要{ protocol_type、land、urgent、num_failed_logins、root_shell 、 num_file_creations 、 num_outbound_cmds 、 is_guest_login 、 same_srv_rate 、 diff_srv_rate 、 dst_host_same_srv_rate 、 dst_host_srv_diff_host_rate}總共32個屬性與不經過縮減的41種屬性來做比較[7]。

實驗結果使用 20 種屬性的準確率為 99.22，相較於 41 種屬性的準確率為 99.25，略降了 0.03，而使用 32 種屬性的準確率和使用 41 種屬性的準確率是相等的，都為 99.25；至於第二部分使用特別的 SVM 特徵等級方法，稱為支援向量決策函數(Support Vector Decision Function ; SVDF)取得的第一重要屬性為 11 個，分別為{duration、src_bytes、dst_bytes、count、srv_count、serror_rate、srv_serror_rate、dst_host_count、dst_host_same_src_port_rate、dst_host_serror_rate、dst_host_srv_serror_rate}，次等重要的屬性 8 個，分別為 {protocol_type 、 service 、 flag 、 hot 、 logged_in 、 same_srv_rate 、 dst_host_srv_count、dst_host_same_srv_rate}，其餘為不重要屬性，實驗表中顯示使用重要的 11 個屬性得到的準確率為 99.16，比原本的 99.25 降低了 0.09，而加上次等重要的屬性應為 19 個，但在原文中卻採用 32 個屬性[7]，故可能有錯誤，因此我們不採用第二部份的屬性，所以在第四章的實驗裡我們將只針對第一部分來分析。

2.3 資料預處理相關研究

由前一節的KDD-Cup資料集相關研究我們可以看出資料預處理對於入侵偵測系統效率的重要性，而目前針對資料做預處理的研究有許多方向，如針對網頁記錄檔、電波訊號或是入侵偵測系統的資料等等，但是這些研究主要目的都是為了使後續工作能夠得到更佳的結果。如N. Zhang等人為了提高資料準備處理速度，利用顧客檢視資料來探勘網頁進行預處理的工作，主要方法是建立顧客檢視的物件圖表分析物件特徵，然後將這些物件歸類，進而藉由將資料整合過後能夠使資料探勘起來更有效率[8]，但如果資料量是龐大的或是多變的，則可能會使整合的資料和原資料沒太大差異，改善品質便會不容易得到好的結果，因此不適用於龐大的、多變的入侵偵測資料。H.Mirghasemi等人提出對腦電波訊號做預處理，他們使用三種過濾資料的方法，為Bandpass Digital Filtering、Median Flitering及Facet Method分別針對三種分類器Fisher Linear Discriminant(FLD)、Kernel Fisher discriment(KFD)、Neural Network(NN) 進行過濾預處理動作，以使三種分類器經過分類過後都能使腦電波的判斷準確率獲得提升[9]，但是目前入侵偵測使用的分類器有許多種類，因此很難針對每一種都得找出一個最適合的過濾預處理方法。Ping-Feng Pai等人提出使用結構風險最小化原理(Structure Rsk Minimization principle)及支援向量機(Support Vector Machines；SVM)時，透過量尺技術(Scaling Techniques)和差分化技術(Differencing Techniques)的預處理方法配合，準確率會比其他預處理方法好[10]，此篇的主要訴求是以增加準確率為重點，但可能因為透過多種技術的結合而增加系統負荷。H.Hannah Inbarani等人是使用迅速修訂(Quickreduct)和簡單的變精度約略集(Variable precision rough Set)去找出網頁記錄檔的多餘屬性資料，藉由刪除多餘的資料，使網頁記錄檔不浪費太多空間[11]，但是並沒有深入探討刪除屬性對日後使用網頁記錄檔的影響程度。

2.4 約略集理論

由上一節預處理的介紹我們可以得知，入侵偵測資料是屬於多變且龐大的，

而我們如何從這些資料裡清楚了解獲得的資訊並且分析它們，便得透過約略集理論來建立訊息系統，並透過約略集理論中的許多方法來分析入侵偵測資料，使我們能夠清楚地看到分類效果。約略集理論在 1982 年時由波蘭 Zdzislaw Pawlak 所提出的數學方法，用來處理含糊與不精確資訊的問題。由於在應用約略集理論進行分析時，不需服從任何的假設，這使得約略集理論的應用彈性更大。而且因為約略集理論可以挖掘出資料屬性彼此之間的關係，這也使得約略集理論被應用在許多領域，例如決策分析(Decision Analysis)、資料庫知識挖掘(Knowledge Discovery from Databases)、專家系統(Expert Systems)、決策支援系統(Decision Support Systems)中的識別模式(Pattern Recognition)等[12]。在本研究中所應用到的約略集理論(Rough Set Theroy)的基礎知識介紹如下:

- 訊息系統(Information System ; IS)
  一個訊息系統 IS 的定義如下

  $$IS = (U, A, f)$$

  其中U為全域（Universe），在我們的入侵偵測資料集中代表每一筆的正常或攻擊的資料訊息，A為屬性集合，每一個屬性$a \in A$（屬性a歸屬於屬性A所考慮的集合），在我們的入侵偵測資料集中代表一筆資料訊息的每一個特徵，這些特徵皆由 KDD-Cup 所添加，最後定義一個訊息函數（Information Function）$f_a : U \to V_a$，其中V_a為屬性a的可能值所構成的集合，稱為屬性a的值域（Domain），所以一個訊息函數表可以藉由每一筆紀錄的所有屬性值建立出來。

- 難以辨識的關係（Indiscernibility Relation）

  對於屬性$B \subset A$的每個集合，難以辨識的關係$Ind(B)$可以下面方式定義之：若$\forall b \subset B$，$b(x_i) = b(x_j)$，其中$b(x_i)$表示物件x_i相對於屬性b的值，也就是屬性集合B中的任一屬性b無法將x_i與x_j兩個物件區別開

來，因此兩物件x_i與x_j藉由屬性B的集合是難以辨識的，x_i與x_j若以屬行集合B來區分是屬於同一類。因此定義$Ind(B)$，我們可以將所有物件分類。$Ind(B)$的同等類別稱為B中的基本集合，因為它表示物件之最小難以辨識的群體。對於U的任何單元x_i，關於$Ind(B)$之x_i的同等類別可表示為$[x_i]_{Ind(B)}$。建構基本集合是約略集中分類的首要步驟。如表 2 基本集合則為x_1, x_5、x_3, x_6及x_2, x_6。

表 2 ： 簡易分類表範例一

U/A	Attr1	Attr2	Attr3
x_1, x_5	1	3	1
x_3, x_6	2	2	2
x_2, x_6	1	3	3

- 上界與下界近似（Lower and upper approximations）

以約略集方法進行資料分析全靠兩個基本觀念，稱之為集合的下界與上界近似（the lower and the upper approximations of a set）（如圖 4），若橢圓形包圍範圍為X集合，其中X若以 KDD-Cup1999 資料集來看是屬於決策屬性集合 D 中某一決策屬性d_i類別的所有資料物件的集合，圖中每一個格子均代表一個物件，則：

- 黑色格子表示”無疑”屬於該X集合，即為下界近似
- 灰色格子表示”可能”屬於該X集合，即為上界近似

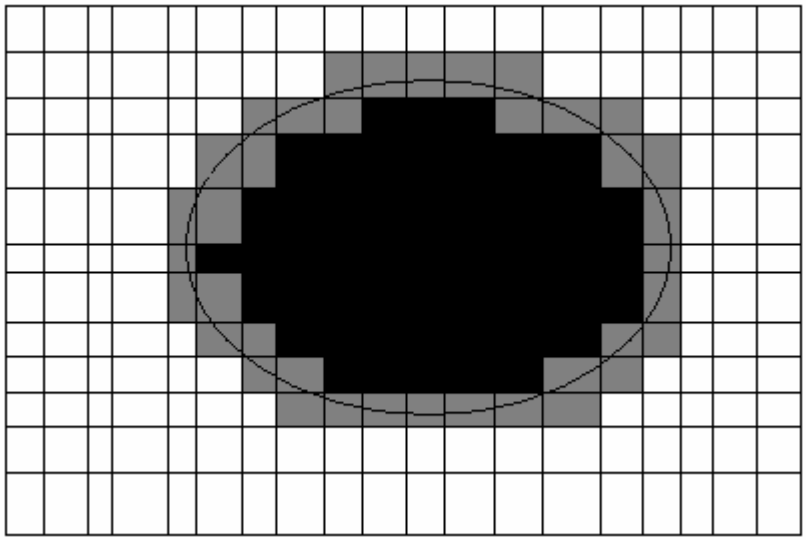

圖 4 ： 上界與下界近似示意圖

令 X 表示全域 U（$X \subseteq U$）單元的子集合，以屬性集合 B（$B \subseteq A$）來分類之狀況下，X 的下界近似表示為 $\underline{BX}$，其定義為所有包含於 X 中的這些基本集合的聯集（Union），一般的表示型式為：

$$\underline{BX} = \left\{ x_i \in U \mid [x_i]_{Ind(B)} \subset X \right\}$$

上面的式子的意義為：集合 X 的下界近似是物件 x_i 的集合，其屬於包含於 X 內的基本集合（於空間 B 內）。

集合 X 的上界近似表示為 $\overline{BX}$，為那些與 X 有非空交集之基本集合的聯集，一般的表示型式為：

$$\overline{BX} = \left\{ x_i \in U \mid [x_i]_{Ind(B)} \cap X \neq 0 \right\}$$

對於 X 的下界近似的任何物件 x_i（亦即：$x_i \in \underline{BX}$），可以確定的是，其必定屬於 X；而對於 X 的上界近似的任何物件 x_i（亦即：$x_i \in \overline{BX}$），我們只能說 x_i 也許屬於 X，但 x_i 隸屬的類別 $[x_i]_{Ind(B)}$ 中一定能找到一個以上的元件，也在 X 中出現，兩者的差別：

$$BNX = \overline{BX} - \underline{BX}$$

稱為U中X的邊界（Boundary）。

若下界與上界近似相同（亦即$\overline{BX} = \underline{BX}$），則集合$X$是可定義的（Definable），否則集合$X$在$U$中就是無法定義的（Undefinable）。在$U$中有四種無法定義的集合類型：

1. 若$\underline{BX} \neq \phi$且$\overline{BX} \neq U$，則稱X在U中是約略定義的（Roughly Definable）；
2. 若$\underline{BX} \neq \phi$且$\overline{BX} = U$，則稱X在U中是表面上無法定義的（Externally Undefinable）；
3. 若$\underline{BX} = \phi$且$\overline{BX} \neq U$，則稱X在U中是內部無法定義的（Internally Undefinable）；
4. 若$\underline{BX} = \phi$且$\overline{BX} = U$，則稱X在U中是完全無法定義的（Totally Undefinable）；

其中ϕ表示空集合。

此外，定義$POS_B(X) = \underline{BX}$稱為$X$的$B$-正區域（$B$-Positive Region），表示可確定分類於集合X中之這些物件的集合；而$NEG_B(X) = U - \underline{BX}$稱為$X$的$B$-負區域（$B$-Negative Region），表示可將物件集合清楚分類為屬於X的餘集（Complement）（或稱不屬於X）；$BN_B(X)$稱為X的B-邊線區域（B-Borderline region），是全域的無法決定（Undecidable）區域，亦即就屬性B而言沒有屬於邊界的物件可確定分類為X或$\overline{X}$ [13]。

三、 RST 入侵偵測預處理機制

由於入侵偵測系統需要處理的資料包含網路訊務及系統稽核資訊，然而這些資料相當龐大，因此如果我們能將許多不必要或是不重要的訊息全部去除，就能夠在不影響偵測的準確性，使得入侵偵測系統效率提升，所以我們希望藉由約略集理論的方法來找出那些資料對於入侵偵測系統偵測攻擊是沒有幫助的，接著將這些資料刪除，以減少入侵偵測系統需要處理的資料量，進而使入侵偵測系統的效能得到提升。本章將於 3.1 節介紹提出的系統架構，其中刪減資料的機制會在 3.2 節介紹，最後 3.3 節我們提出兩個基於我們資料量縮減後可應用於入侵偵測系統的構想。

3.1 RST 預處理架構及運作方式

本文提出的 RST 入侵偵測系統架構，如圖 5，其中虛線框起的部分為我們所提出的 RST 入侵偵測預處理機制，我們的研究是使用 KDD-Cup 1999 資料集，經由 20/80 法則[14]，取得訓練資料集和測試資料集，並藉由約略集理論(Rough Set Theory；RST)建立訊息系統，然後使用訓練集資料進行兩階段的屬性縮減(Attribute Reduct)的動作，縮減後我們將所得到縮減後的必需屬性結合到 IDS 的預處理機制(IDS Preprocessing Scheme)， 使其可以經由必要屬性來減少資料量後再傳送給入侵偵測系統判斷。

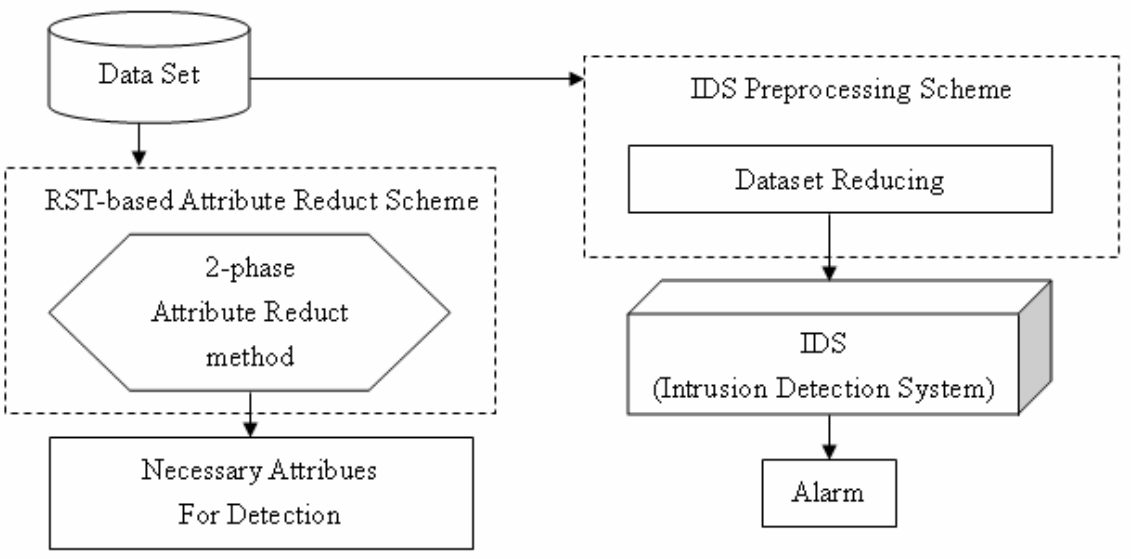

圖 5 ： RST 入侵偵測系統架構圖

而 RST 入侵偵測預處理架構中的二階段屬性縮減方法如圖 6，我們使用的資料集為 KDD-Cup 1999 入侵偵測資料集，為了實驗方便以及更加快速，因此我們將原本龐大的資料以隨機選的方式選出 10 萬筆資料，然後我們利用約略集研究系統將 10 萬筆資料建立具有決策屬性的訊息系統，接著我們使用 20-80 法則將 10 萬筆資料分割成分別為 2 萬筆的訓練集資料和 8 萬筆的測試集資料，然後由訓練資料經過第一階段和第二階段的兩階段屬性縮減，來獲得屬性縮減後的屬性，接著我們使用縮減後的屬性數和未縮減屬性前的資料集做比較，然後評估是否縮減屬性後會影響其準確及覆蓋率，如果我們發現縮減後的資料量明顯使得入侵偵測系統的準確率或覆蓋率下降，我們會再進行修正，最後使用測試集資料來做最後的比較，如果導致準確率或覆蓋率降低，我們便在回饋修正，直到準確率及覆蓋率能夠穩定，而圖 5 中使用到的屬性縮減理論將於 3.2 節中逐一介紹。

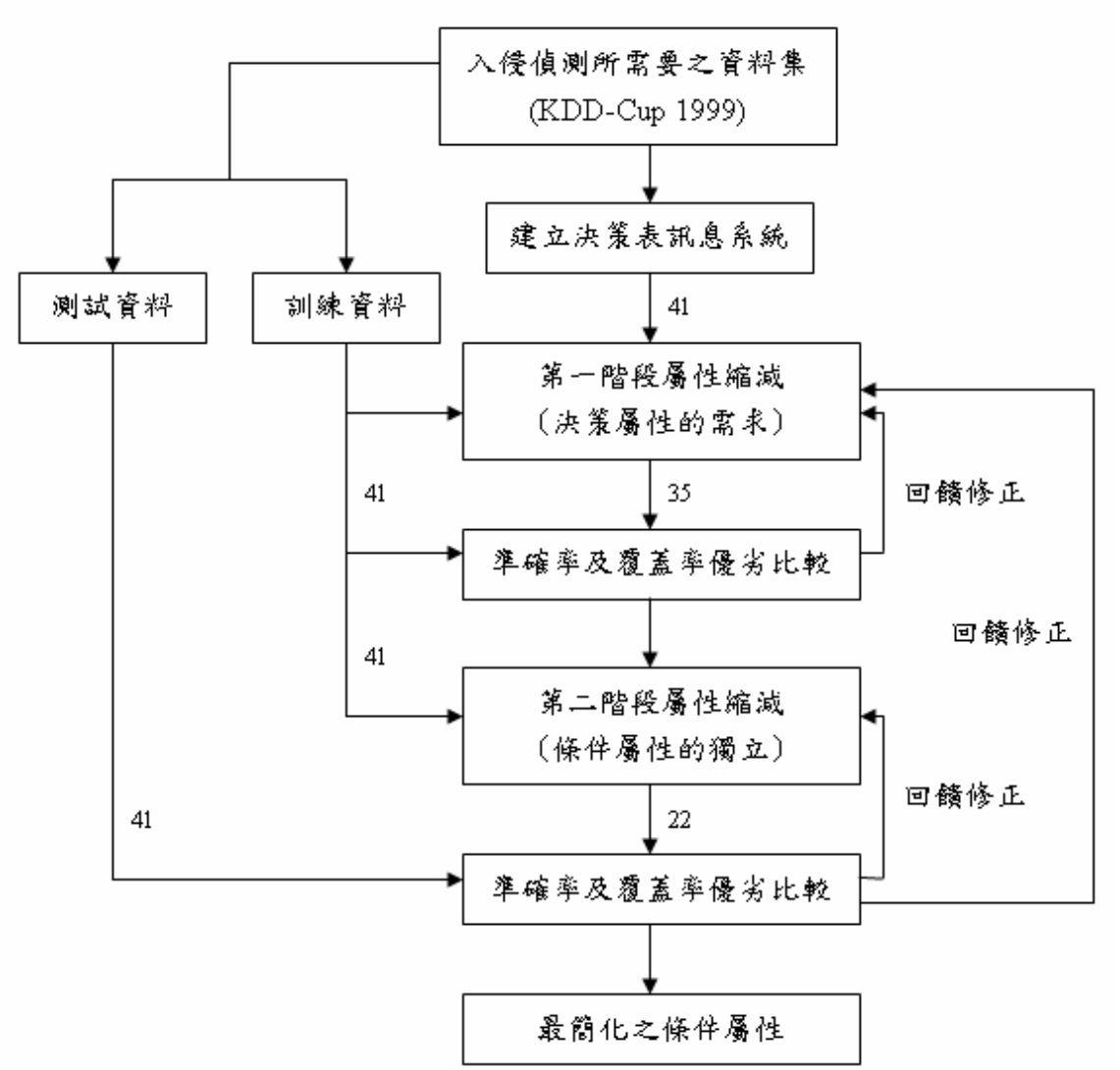

圖 6 ： RST 二階段屬性縮減方法

3.2 二階段屬性縮減

首先根據取得的資料集建立一個訊息系統 IS(Information System)。根據式(1)再建立出一個具有決策屬性的訊息系統，在此稱為決策表訊息系統，由 KDD-Cup 1999 資料集建立出來的訊息系統如圖 7，因此我們可以透過具有標記決策屬性的訊息系統來判斷我們偵測的準確率。

條件屬性

20000...	duration	service	src_bytes	dst_bytes	wrong_fragment
0:1	0	http	232	2654	0
0:2	0	http	288	1046	0
0:3	0	http	345	6055	0
0:4	0	http	164	2112	0
0:5	0	http	273	4471	0
0:6	0	smtp	3446	321	0
0:7	0	http	297	770	0
0:8	0	http	207	376	0
0:9	0	private	0	0	0
0:10	0	http	287	1572	0
0:11	0	private	0	0	0
0:12	0	http	207	1918	0
0:13	0	private	0	0	0
0:14	0	domain_u	43	76	0
0:15	0	private	0	0	0
0:16	0	http	270	9590	0
0:17	0	http	0	0	0
0:18	0	ftp_data	383	0	0
0:19	0	http	248	1249	0
0:20	0	other	146	0	0

...

條件屬性　　決策條件屬

dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	attr42
0	0	0	normal.
0	0	0	normal.
0.01	0	0	normal.
0	0	0	normal.
0	0	0	normal.
0	0	0	normal.
0	0	0.01	normal.
0	0	0	normal.
1	0	0	neptune.
0	0	0	normal.
1	0	0	neptune.
0	0	0	normal.
1	0	0	neptune.
0	0	0	normal.
1	0	0	neptune.
0	0	0	normal.
0	1	1	normal.
0	0	0	normal.
0	0	0	normal.
0	0	0	normal.

圖 7 ： 訊息系統範例

$$IS = (U, A = C \cup D, f) \tag{1}$$

式(1)中 C 為條件屬性(Condition Attribute)，如 duration、service、src_bytes 等，屬性裡面的訊息可以用來幫助探勘是否為攻擊。D 為決策屬性，如 normal、neptune 等，在 KDD-Cup 1999 資料集裡為第 42 個屬性，用來得知物件的分類項目，$C \cup D$ 為全部屬性 A (Decision Attribute)，因此我們可以藉由 U 中所有記錄來觀察與決策屬性值的關係，並找出相關規則，藉由這些規則來將 U 歸類，以便得知每一筆資料是屬於正常資料或某種攻擊行為，之後針對 IS 架構進行兩階段屬性縮減的處理，以下將針對兩階段屬性縮減方法介紹。

3.2.1 第一階段屬性縮減-依決策屬性

因為決策屬性內包含數種決策類別，但是每一種類別用來進行判斷的條件屬性 C 也都不相同，所以如果我們找出並刪除無助於將資料集 U 順利依據決策屬性 D 分類的條件屬性，便能夠達到屬性縮減的效果，因此我們在第一階段先根據決策屬性來將 KDD-Cup1999 資料集分類，每個決策屬性類別除正常狀況(normal)外均分別表示某一攻擊手法，然後我們利用每個決策屬性所需條件屬性都不相同的特性，找出判斷每個決策屬性的必要條件屬性有哪些，然後針對沒有幫助判斷的條件屬性進行刪減的動作。

對於第一階段屬性縮減的方法，可以用表 3 的例子來解釋，依據四個屬性 Attr1 、 Attr2 、 Attr3 、 Attr4 可以將所有資料 $U = \{x1, x2, x3, x4, x5, x6, x7, x8, x9, x10\}$ 分類成五個類別，我們可以很容易的由表中的第一筆及第三筆發現，要找出 normal 的物件可以使用 Attr1，若值為 1 則表示該類型為 normal，也可以使用 Attr3，若值為 1 或 3 則為 normal 類型，

由第二筆及第四筆資料知道要找 smurf 的物件可以使用 Attr2，若 Attr2 值為 2 則表示為 smurf 類型，Attr3 值若為 2 也為 smurf 類型，而要找出 pod 攻擊的物件可以由第五筆知道 Attr1 值為 3、Attr2 值為 1 及 Attr3 值為 4 均為 pod 類型，都有有助於我們找出決策類別，而 Attr4 卻一點幫助都沒有，因此我們可以將 Attr4 刪除，所以我們第一階段的屬性縮減便是希望透過這種方式找出對於決策屬性的分類沒有幫助的條件屬性有哪些，進而刪除他們，以達到屬性初步縮減的目的。

表 3 ： 簡易分類表範例二

U/A	Attr1	Attr2	Attr3	Attr4	D
$\{x1, x3, x9\}$	1	3	1	1	normal
$\{x2, x7, x10\}$	2	2	2	1	smurf
$\{x4\}$	1	3	3	1	normal
$\{x5, x8\}$	2	2	2	1	smurf
$\{x6\}$	3	1	4	1	pod

根據 KDD-Cup 1999 資料集將決策屬性 D 分為 19 種決策類別，因此我們將決策屬性的每一個決策類別形成集合，記為:

$$D_i = \{d_1, d_2, d_3, \Lambda\ , d_{19}\} \tag{2}$$

首先我們針對所有決策屬性 D 找出具有影響的條件屬性集合 C_d，使得 $Ind(C_d) = Ind(C)$，亦即以 C_d 來分類的結果與原來用 C 來分類的結果相同，其中 $C_d \subseteq C$。

我們取 C_d 的補集 A'，即為我們應刪除的屬性，如下式:

$$A' = C - C_d \tag{3}$$

在針對所有的決策類別進行判斷後，再分別針對每一個決策類別進行判斷，以達到更精確的效果，因此我們為每一個決策屬性d_i找出具有影響程度的C_i，使得$Ind(C_i) = Ind(C_d) = Ind(C)$

其中，C_i定義為$C_i = \{\{C_{i1}, C_{i2}, \text{K } C_{ij}\} | 1 \le j \le 41, C_i \subseteq C_d\}$

故針對各別決策屬性d_i分析後所需刪除的屬性集合A''為:

$$A'' = C - \bigcup_{1 \le i \le 19} \{C_i\} \tag{4}$$

原來 RST 縮減方法應為$A' \cup A''$，但將於第四章說明，我們於是改取A'和A''的交集，即為所有無益於U分類的條件屬性，

$$\overline{C} = A' \cap A'' \tag{5}$$

所得$\overline{C}$即為我們第一階段所應刪減的屬性。

3.2.2 第二階段屬性縮減-依條件屬性

對於第二階段屬性縮減的方法可以由表 4 的例子來說明，我們可以看到 Attr1 能夠分辨出 normal，Attr2 能夠分辨出 smurf，Attr4 能夠分辨出 pod 及 portsweep，若我們不使用 Attr4 則表 4 中的第三筆及第四筆的所有屬性值會完全相同，所以會分成同一類，分類結果將如表 5，portsweep 和 pod 將無法分辨清楚正確類別，因此可以看出每個屬性將資料分類成決策類別的重要性，但是若我們能夠找到刪除屬性後對於分類成決策屬性並不會產生影響的屬性，那表示這個屬性是多餘的，如表 6，我們將 Attr3 刪除後所得到的分類效果一樣，因此我們第二階段的屬性縮減便希望藉由這種方法來找出對於決策類別沒有影響的條件屬性，然後藉由刪除它們以達到屬性縮減的目的。

表 4 ： 簡易分類表範例三

U/A	Attr1	Attr2	Attr3	Attr4	D
$\{x1, x3, x9\}$	1	1	1	2	normal
$\{x2, x7, x10\}$	2	2	2	2	smurf
$\{x4\}$	2	1	3	3	pod
$\{x5, x8\}$	2	1	3	4	portsweep
$\{x6\}$	2	2	1	2	smurf

表 5 ： 簡易分類表範例四

U/A	Attr1	Attr2	Attr3	D
$\{x1, x3, x9\}$	1	1	1	normal
$\{x2, x7, x10\}$	2	2	2	smurf
$\{x4, x5, x8\}$	2	1	3	pod, portsweep
$\{x6\}$	2	2	1	smurf

表 6 ： 簡易分類表範例五

U/A	Attr1	Attr2	Attr4	D
$\{x1, x3, x9\}$	1	1	2	normal
$\{x2, x7, x10\}$	2	2	2	smurf
$\{x4\}$	2	1	3	ipsweep
$\{x5, x8\}$	2	1	4	portsweep
$\{x6\}$	2	2	2	pod

因此在第二階段屬性刪減的方法上，我們先取得第一階段刪減屬性集合 $\overline{C}$ ，並取 $\overline{C}$ 相對於 C 的補集:

$$\Phi = C - \overline{C} \tag{6}$$

接著使用Φ來繼續進行第二階段的刪減屬性。

在第二階段屬性縮減方法中若能得知在Φ中的每一個單獨的條件屬性C_k對每一個決策屬性d_i的影響程度有多少，便能夠知道屬性C_k的重要程度，因此可以用以決定是否刪除屬性C_k。

我們找出個別去除Φ內的屬性C_k對於U來判斷決策屬性的分類不會有影響的屬性集合，如下式:

$$POS_{\Phi}(U) = POS_{(\Phi - C_k)}(U) \tag{7}$$

式(7)表示，屬性Φ集合中使用屬性C_k前和去除後所得到的資料訊息對於分類效果並沒有改變，則表示屬性C_k是多餘的，$POS_{\Phi}(U)$為前述的下界近似，亦稱為D的Φ-正區域(Φ-Positive Region)，否則屬性C_k在Φ集合中就是不可或缺的，最後得出

$$\Phi - \bigcup_{C_k \in \Phi}\{C_k\} = C - \left\{\overline{C} + \bigcup_{C_k \in \Phi}\{C_k\}\right\} \tag{8}$$

經由第二階段條件屬性的獨立，我們可以找出所有條件屬性對於決策屬性的影響程度，我們藉由分析這些影響程度來找出可以刪除的條件屬性，然後刪除它們，即能夠達到我們屬性刪減的目的。

3.3 IDS 擴充版本

根據以上的方法，我們縮減 KDD-Cup 1999 資料集，並得到 22 個屬性，其中 8 個屬性為判斷 normal 必須屬性，而 18 個為判斷所有攻擊類型所需屬性，而相關實驗分析將於第四章介紹，於此我們僅依據縮減得出針對圖 5 的 IDS 擴充概念。

1. 擴充版本一：

根據上一節中的第二階段屬性縮減處理中，我們可以取得每一個屬性對於決策類別的影響程度，因此我們可以找出會影響 normal 類別的條件屬性，也就是說透過這些屬性我們能夠找出所有屬於 normal 類型的物件，其餘的物件則為異常，而剩下的物件我們可以針對我們有興趣的決策屬性在仔細分類，擴充版本一架構圖如圖 8。其中 8 個屬性針對偵測正常狀的的屬性，18 個為只判別異常部分的決策類別，不包含正常部分，因此在預處理部份我們先使用判斷正常的屬性來將資料進行簡單分類，先選擇正常資料，進而減少 IDS 所需判斷的資料量，使用 18 個屬性來找出可疑的異常資料。

而擴充版本一也可讓我們針對有興趣的攻擊來判斷，我們可以由表 7 來解釋，表中可以看到使用 Attr1 可以將 normal 類別找出來，因此我可以得知 $\{x1, x3, x4, x5, x8, x9\}$ 都是正常的物件，其餘 $\{x2, , x6, x7, x10\}$ 為異常，而若我們對 pod 物件有興趣，則我們可以透過 Attr3 再將異常部分的物件進行判斷，則我們便能找出 $\{x6\}$ 為 pod，因此透過這個方法我們能夠使入侵偵測系統對於興趣知道的決策屬性挑選必要的條件屬性來偵測，便能夠使入侵偵測系統的效能再提高。

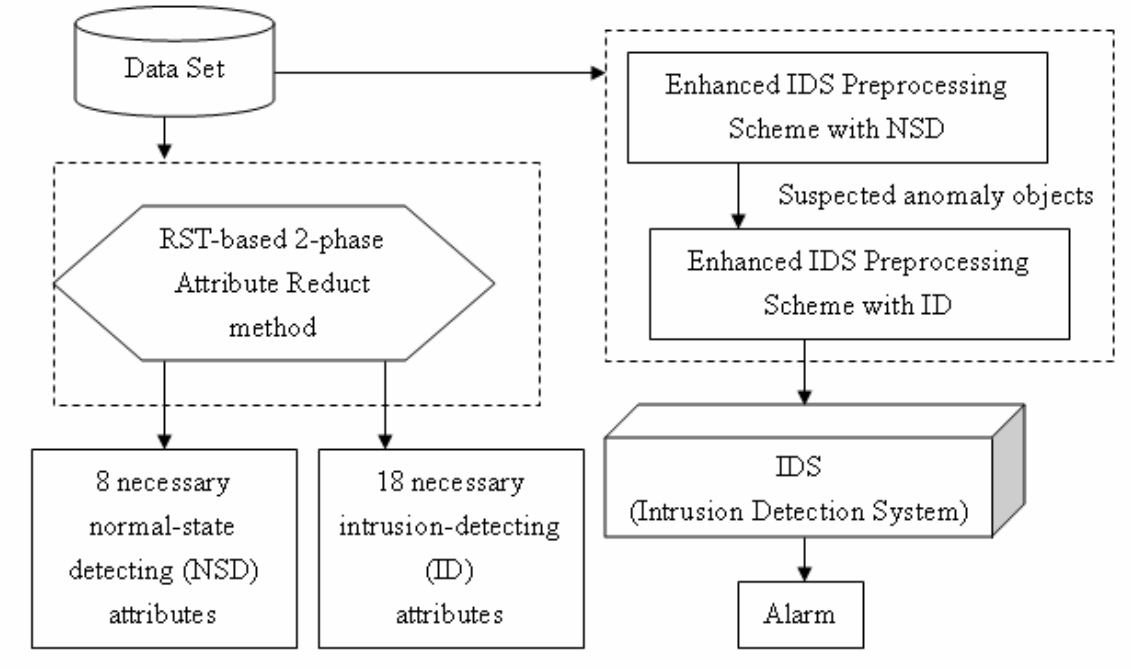

圖 8 ： RST 入侵偵測系統擴充版本一架構圖

表 7 ： 簡易分類表範例六

U/A	Attr1	Attr2	Attr3	D
$\{x1, x3, x9\}$	2	3	1	normal
$\{x2, x7, x10\}$	3	1	1	smurf
$\{x4\}$	2	3	1	normal
$\{x5, x8\}$	2	3	1	normal
$\{x6\}$	3	3	2	pod

2. 擴充版本二 ：

根據上面兩個屬性縮減的處理後，我們發現我們由約略集理論進行分類的方法會產生一個問題，以下舉個例子敘述，如表 8，若我們使用三個屬性將資料分類後，若資料 $\{x1, x3, x4, x5, x9\}$ 屬於 normal 類別，依據我們使用的

約略集理論下近似我們可以得知，{*x*1,*x*3,*x*9} 、{*x*4} 皆能夠分類出來，但是{*x*5}卻無法偵測，因為分類結果{*x*5,*x*8}為一類，若我們使用的是上近似法便能夠將{5}也偵測到，但是卻會連帶的將{*x*8}也分類為 normal 類別，因此如果{*x*8}為某種攻擊變會造成誤判，所以我們不考慮使用上近似法，但是使用下近似法卻會造成資料遺失無法判別。

表 8 ： 簡易分類表範例七

U/A	Attr1	Attr2	Attr3
{*x*1,*x*3,*x*9}	2	1	3
{*x*2,*x*7,*x*10}	3	2	1
{*x*4}	2	2	3
{*x*5,*x*8}	1	1	4
{*x*6}	1	1	2

由上面提到的問題，我們可以得知使用約略集理論中的下近似所獲得的分類效果會導致某些物件的遺失，也就是無法判斷類別，以致無法分類的物件，因此我們提出一個能夠解決的方法，即圖 5 的第二個擴充版本，我們利用縮減後所得到的 normal 條件屬性來對 IDS 無法判斷的資料進行再次比對，可以得知若經由 8 個 NSD 屬性判別後如果不為 normal 類別，入侵偵測系統則將該筆物件當為攻擊，如此剩餘無法判斷的攻擊只有屬於攻擊類型的資料以及 normal 類型無法判斷的資料會被阻擋在入侵偵測系統外。因此可以看出來此方法的缺點，雖然能夠偵測到剩餘的所有攻擊，但也會有部份 normal 類型的資料被誤判為攻擊。所以我們的擴充版本二架構圖如圖 9，其中經過二階段屬性縮減後所得兩部份縮減後的資料，8 個屬性針對偵測正常狀態的屬性，22 個能夠偵測所有決策類別的屬性，希望藉由判斷 normal 類別的處理將無法判別的剩餘資料逐一判別，分類成”正常”與”異常”，並再藉由入侵偵測系統阻擋異常類別的資料。

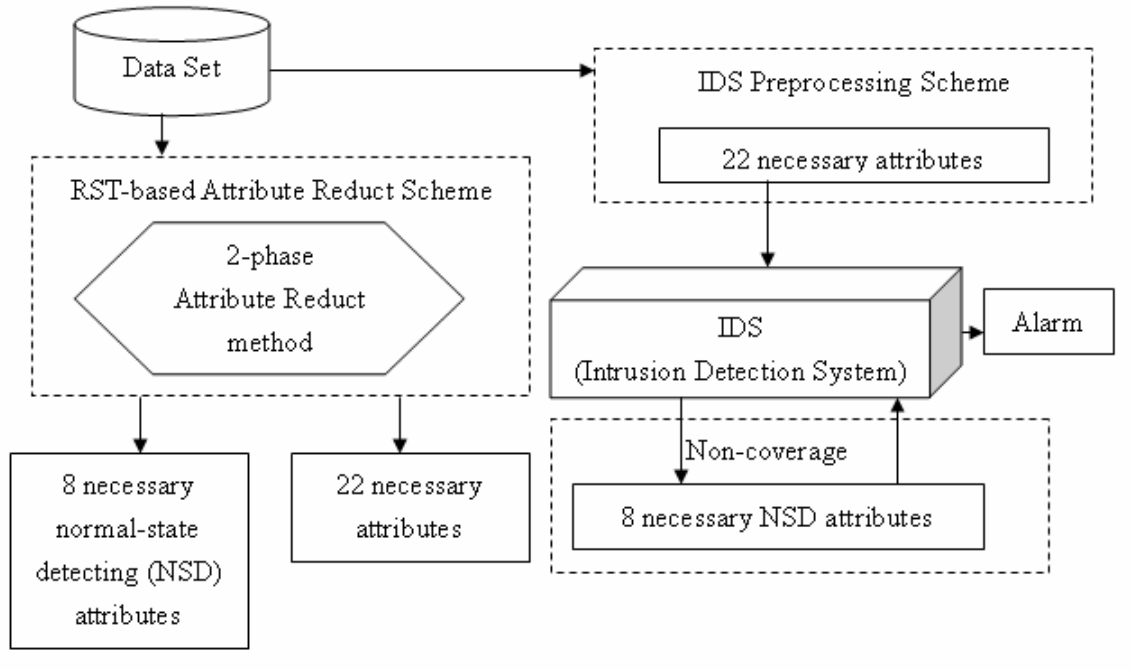

圖 9 ： RST 入侵偵測系統擴充版本二架構圖

四、 實驗結果與分析

本章將介紹我們的各種實驗與分析比較，首先 4.1 節介紹我們的實驗環境，4.2 節使用二階段屬性縮減來找出最佳的條件屬性集合和補償機制所應該用的條件屬性集合，並證明我們所使用的最佳條件屬性集合確實能夠達到減少屬性量卻不影響準確率及覆蓋率。4.3 節介紹我們使用不同 IDS 的分類方法來進行實驗，比較我們二階段屬性縮減後所得到的條件屬性於不同的分類方法的適用狀況，4.4 節我們比較其他使用 KDD-Cup 1999 的相關研究，證明所獲得的二階段縮減條件屬性集合在準確率及覆蓋率均優於其他研究。

4.1 實驗環境

本研究採用約略集研究系統(RSES)[15]來進行實驗，此套軟體由波蘭華沙大學以 Professor Andrzej Skowron 為主的研究團隊所發展出來的。該研究團體根據約略集理論研發此套分析軟體，而我們也使用目前最新版本 2.2.2 版，該軟體可讓使用者在 Windows 的環境下進行分析，是一套親和力極佳的軟體，同時也是一套免費的開放軟體，系統初始介面如圖 10。

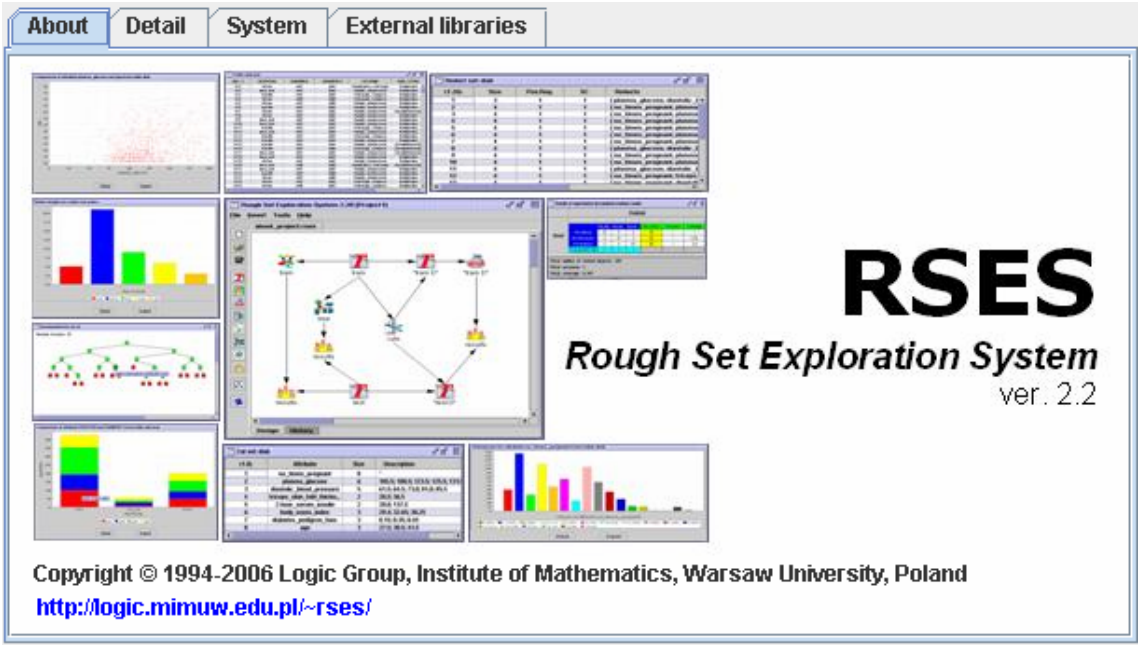

圖 10 ： RSES 系統初始介面

我們可以使用這套軟體來幫助我們進行一些約略集的處理動作，如將資料集建立成訊息系統，然後隨機分割成訓練資料集及測試資料集，我們也可以透過系統裡的分類方法來幫助我們進行比較的動作，如 Decomposition Tree、類神經網路及 KNN，因此這套軟體可以使我們更加方便的進行屬性縮減的研究，而使用這套軟體來幫助進行研究的也不少，如南台科技大學的賴威利先生使用來進行預測燒燙燒患者死亡率的研究[16]、華梵大學的陳士杰用來對理財促銷資料集進行分類[17]及逢甲大學的施奕良使用來遙測影像水稻田知識庫分類[18]等等。而此套軟體也曾經發表過許多文章[19][20][21]，來介紹他們獲得 RESE 方法的想法，其它文章均可在 RESE 網站中取得[15]。而執行此版本的要求配備如下:

- CPU-Pentium 200 MHz
- 128 MB RAM
- Java Virtural Machine version1.4.1

  MS Windows 9x/NT/2000/Me/XP 或是 Linux/i386

本研究的實驗的環境如表 9:

表 9 ： 實驗環境

硬體設備	CPU	Intel core2 1.66G
	記憶體	1.50G
軟體設備	作業系統	Window XP Home edition Sp2
	應用軟體	約略集研究系統 RSES2 (Rough Set Exploration System2.2.2)
		Java Virtural Machine version1.5.1

本研究使用的資料集為 KDD-Cup1999 的資料集資料集中有 42 個屬性，包含 41 個條件屬性和 1 個決策屬性。

4.2 二階段屬性縮減結果

在二階段屬性縮減我們會依據準確率及覆蓋率來決定所需條件屬性。評估準則定義如下:

Accuracy = (O- F-L) / (O - L)

Coverage = (O - L) / O

其中 Accuracy 是準確率，表示利用對訊息系統所找出的規則探勘結果的準確率，Coverage 是覆蓋率，表示過程中所能識別的物件比率，O 表示所有的物件筆數，F 表示誤判物件筆數，L 表示無法判斷的物件筆數，因此準確率的算法是所有物件筆數扣除誤判物件筆數和遺失物件筆數，即為判斷正確的數目，然後除上所有物件扣除遺失筆數；而覆蓋率的算法則為所有物件的筆數扣除遺失的筆數，

即為能夠判別的筆數，然後除上所有的物件數目。

經由第一階段屬性縮減處理的實驗結果如表 10，由表 10 的第一筆紀錄可以得知對於決策類別全部有影響的條件屬性，因此對於所有決策類別沒有影響的條件屬性為 A' ={protocol_type 、 logged_in 、 root_shell 、 num_outbound_cmds 、 is_host_login、is_guest_login}，以及經由前述式(4)分別找出對各個決策類別沒有影響的條件屬性 A'' = { protocol_type、wrong_fragment、 logged_in、root_shell、num_outbound_cmds、is_host_login、is_guest_login }，我們再將這個兩集合取其聯集即為式(6) $\overline{C} = A' \cup A''$，得到 $\overline{C}$ = { protocol_type、wrong_fragment、logged_in、root_shell、num_outbound_cmds、is_host_login、is_guest_login }，但是經由圖 11 及圖 12 所得實驗結果，我們發現多刪除 wrong_fragment 屬性後將導致 teardrop 的覆蓋率由 0.988 降為 0.105，所以 wrong_fragment 資訊對 teardrop 攻擊手法為一重要判斷依據，事實上我們由 teardrop 攻擊手法[22]也可以得知 teardrop 會製造出很多的 fragments，因此我們改取兩條件屬性集合的交集 $\overline{C} = A' \cap A''$，再重新進行實驗，發現準確率及覆蓋率保持穩定，因此我們得到第一階段兩個步驟處理後所得到所能刪減的屬性即為 $\overline{C}$ 。

表 10 ： 經過第一階段處理後所得結果

決策類別(d_i)	需要的條件屬性(C_j)
All(D)	duration、service、flag、src_bytes、dst_bytes、land、wrong_fragment、urgent、hot、num_failed_logins、num_compromised、su_attempted、num_root、num_file_creations、num_shells、num_access_files、count、srv_count、serror_rate、srv_serror_rate、rerror_rate、srv_rerror_rate、same_srv_rate、diff_srv_rate、srv_diff_host_rate、dst_host_count、dst_host_count、

	dst_host_same_srv_rate、dst_host_diff_srv_rate、 dst_host_same_src_port_rate、dst_host_srv_diff_host_rate、 dst_host_serror_rate、dst_host_srv_serror_rate、 dst_host_rerror_rate、dst_host_srv_rerror_rate
Normal(d_1)	duration、service、flag、src_bytes、dst_bytes、hot、 num_failed_logins、num_compromised、su_attempted、num_root、 num_file_creations、num_shells、num_access_files、count、 srv_count、srv_serror_rate、rerror_rate、srv_rerror_rate、 same_srv_rate、diff_srv_rate、srv_diff_host_rate、dst_host_count、 dst_host_count、dst_host_same_srv_rate、dst_host_diff_srv_rate、 dst_host_same_src_port_rate、dst_host_srv_diff_host_rate、 dst_host_serror_rate、dst_host_srv_serror_rate、 dst_host_rerror_rate、dst_host_srv_rerror_rate
neptune(d_2)	service、count、serror_rate、srv_serror_rate、same_srv_rate、 diff_srv_rate、dst_host_srv_serror_rate
satan(d_3)	service、count、serror_rate、rerror_rate、same_srv_rate、 diff_srv_rate、dst_host_diff_srv_rate、dst_host_serror_rate、 dst_host_rerror_rate
Teardrop(d_4)	service、srv_count、same_srv_rate
back(d_5)	src_bytes、dst_bytes、rerror_rate、srv_rerror_rate、 dst_host_rerror_rate、dst_host_srv_rerror_rate
warezclient (d_6)	duration、src_bytes、dst_bytes、hot、dst_host_srv_diff_host_rate
portsweep(d_7)	duration、flag、serror_rate、rerror_rate、diff_srv_rate、 dst_host_same_src_port_rate、dst_host_serror_rate、 dst_host_srv_serror_rate、dst_host_rerror_rate
ipsweep(d_8)	src_bytes、dst_bytes、dst_host_srv_diff_host_rate
smurf(d_9)	count、srv_count
Buffer_overflow (d_{10})	duration、src_bytes、dst_bytes
guess_passwd	src_bytes、dst_host_srv_serror_rate

(d_{11})	
nmap(d_{12})	dst_host_diff_srv_rate、dst_host_serror_rate
Warezmaster (d_{13})	dst_bytes
pod(d_{14})	service、src_bytes
ftp_write(d_{15})	dst_bytes、urgent
land(d_{16})	land、dst_host_srv_serror_rate
imap(d_{17})	src_bytes、dst_bytes、num_compromised、num_root、srv_serror_rate、srv_diff_host_rate、dst_host_serror_rate、dst_host_srv_serror_rate
multihop(d_{18})	dst_bytes
rootkit(d_{19})	dst_bytes

	normal.	neptu...	satan.	teardr...	back.	warezc...	portsw...	ipsw...	smurf.	buffer...	guess...	nmap.	warezm...	pod.	ftp...	land.	imap.	multih...	rootkit.	No. of obj.	Accuracy	Coverage
normal.	14,169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14,545	1	0.974
neptune.	0	2,436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4,468	1	0.545
satan.	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	1	0.769
teardrop.	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	1	0.988
back.	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	1
warezclient.	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0.464
portsweep.	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	66	1	0.606
ipsweep.	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	106	1	0.17
smurf.	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	83	1	0.735
buffer_overflow.	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	6	1	1
guess_passw...	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	15	1	0.4
nmap.	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	43	1	0.186
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	6	1	1
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	24	1	1
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	1	1
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	4	1	1
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	1	1
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

圖 11 ： 未刪除任何屬性前所得結果 1

	normal.	neptu...	satan.	teardr...	back.	warezc...	ports...	ipswe...	smurf.	buffer...	guess...	nmap.	warezm...	pod.	ftp_...	land.	imap.	multih...	rootkit.	No. of obj.	Accuracy	Coverage
normal.	14,169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14,545	1	0.974
neptune.	0	2,436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4,468	1	0.545
satan.	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	1	0.769
teardrop.	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	1	0.105
back.	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	1
warezclient.	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0.464
portsweep.	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	66	1	0.606
ipsweep.	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	106	1	0.17
smurf.	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	83	1	0.735
buffer_overflow.	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	6	1	1
guess_pass...	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	15	1	0.4
nmap.	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	43	1	0.186
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	6	1	1
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	24	1	1
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	1	1
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	4	1	1
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	1	1
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

圖 12 ： 刪除七個屬性後所得結果

經由第一階段屬性縮減後我們由式(5)取得$\overline{C}$為我們應刪除的屬性，因此我們第二階段則由式(6)取得剩餘的屬性Φ ，繼續執行第二階段的屬性縮減處理。

因此我們由表 11 可以得知，每個條件屬性對於各種攻擊類別的影響程度，如 dauration，若沒有使用這個屬性則受影響的決策類別有三個分別為 warezclient、portsweep、buffer_overflow，而影響程度可以由圖 13 及圖 14 看出雖然準確率沒有影響但是 warezclient 覆蓋率由 0.464 降為 0.453、portsweep 覆蓋率由 0.606 降為 0.258 、buffer_overflow 覆蓋率由 1 降為 0.833；若我們以 dst_bytes 來看就更加明顯，由表 11 可以看到受影響的分別有 normal、ipsweep、buffer_overflow、warezmaster、ftp_write、multihop、rootkit 七個決策類別，而影響程度我們可以從圖 13 及圖 15 看出 warezmaster、multihop、rootkit 準確率及覆蓋率由 1 降為 0，normal 覆蓋率由 0.974 降為 0.964、ipsweep 覆蓋率由 0.17 降為 0.16、buffer_overflow 覆蓋率由 1 降為 0.667、ftp_write 覆蓋率由 1 降為 0.333。因此我們可以找出每種屬性對於決策類別的影響，進而去判斷是否保留屬性。

	normal.	neptu...	satan.	teard...	back.	ware...	ports...	ipsw...	smurf.	buffer...	gues...	nmap.	ware...	pod.	ftp_...	land.	imap.	multi...	rootkit.	No. of obj.	Accuracy	Coverage
normal.	14,169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14,545	1	0.974
neptune.	0	2,436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4,468	1	0.545
satan.	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	1	0.769
teardrop.	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	1	0.988
back.	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	1
warezclient.	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0.464
portsweep.	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	66	1	0.606
ipsweep.	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	106	1	0.17
smurf.	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	83	1	0.735
buffer_overflow.	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	6	1	1
guess_passwd.	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	15	1	0.4
nmap.	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	43	1	0.186
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	6	1	1
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	24	1	1
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	1	1
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	4	1	1
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	1	1
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

圖 13 ： 未刪除任何屬性前所得結果 2

	normal.	neptune.	satan.	teardr...	back.	warez...	ports...	ipsw...	smurf.	buffer...	guess...	nmap.	warez...	pod.	ftp_w...	land.	imap.	multih...	rootkit.	No. of obj.	Accuracy	Coverage
normal.	14,169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14,545	1	0.974
neptune.	0	2,436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4,468	1	0.545
satan.	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	1	0.769
teardrop.	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	1	0.988
back.	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	1
warezclient.	0	0	0	0	0	82	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0.453
portsweep.	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	66	1	0.258
ipsweep.	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	106	1	0.17
smurf.	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	83	1	0.735
buffer_overflow.	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	6	1	0.833
guess_pass...	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	15	1	0.4
nmap.	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	43	1	0.186
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	6	1	1
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	24	1	1
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	1	1
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	4	1	1
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	1	1
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

圖 14 ： 刪除條件屬性 dauration 後所得實驗結果

	normal.	neptu...	satan.	teard...	back.	warez...	ports...	ipswe...	smurf.	buffer...	guess...	nmap.	warez...	pod.	ftp_...	land.	imap.	multi...	rootkit.	No. of obj.	Accuracy	Coverage
normal.	14,020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14,545	1	0.964
neptune.	0	2,436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4,468	1	0.545
satan.	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	1	0.769
teardrop.	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	1	0.988
back.	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	1
warezclient.	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0.464
portsweep.	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	66	1	0.606
ipsweep.	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	106	1	0.16
smurf.	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	83	1	0.735
buffer_overflow.	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	6	1	0.667
guess_passw...	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	15	1	0.4
nmap.	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	43	1	0.186
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	24	1	1
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	3	1	0.333
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	4	1	1
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	1	1
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

圖 15 ： 刪除條件屬性 dst_bytes 後所得實驗結果

由上面敘述，我們可以透過式(7)可以得到表 11 來，我們再藉由觀察表 11 來得知所有受到影響的攻擊類別，還有它們分別受到的影響程度，所以我們由表中可以很清楚的得知{duration, service,src_bytes,dst_bytes,land,wrong_fragment,hot, count,srv_count,serror_rate,rerror_rate,same_srv_rate,diff_srv_rate,dst_host_count,dst_host_srv_count,dst_host_diff_srv_rate,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate}，這些屬性對於決策類別都是有影響的，而由於入侵偵測系統是希望能夠使所有的攻擊都能夠偵測到，因此只要我們發現移除屬性對於攻擊類別有影響，也就是說移除屬性會使得準確率或覆蓋率降低，我們便會採取保留的動作，所以整理過後我們可以發現在第二階段的屬性縮減，我們可以移除的屬性集合為下:{ protocol_type,flag,urgent,num_failed_logins,logged_in,num_compromised, root_shell,su_attempted,num_root,num_file_creations,num_shells,num_access_files,num_outbound_cmds,is_host_login,is_guest_login,srv_serror_rate,srv_rerror_rate,srv_diff_host_rate,dst_host_same_srv_rate}。

表 11 ： 經過第二階段處理後所得結果

屬性	受影響的攻擊類別	影響程度	
duration	warezclient	Accuracy	1→1
		Coverage	0.464→0.453
	portsweep	Accuracy	1→1
		Coverage	0.606→0.258
	buffer_overflow	Accuracy	1→1
		Coverage	1→0.833
service	normal	Accuracy	1→1
		Coverage	0.974→0.973

	neptune	Accuracy	1→1
		Coverage	0.545→0.512
	pod	Accuracy	1→1
		Coverage	1→0.958
src_bytes	normal	Accuracy	1→1
		Coverage	0.974→0.956
	back	Accuracy	1→1
		Coverage	1→0.994
	warezclient	Accuracy	1→1
		Coverage	0.464→0.403
	ipsweep	Accuracy	1→1
		Coverage	0.17→0.047
	buffer_overflow	Accuracy	1→1
		Coverage	1→0.833
	guess_passwd	Accuracy	1→1
		Coverage	0.4→0.333
	pod	Accuracy	1→1
		Coverage	1→0.042
dst_bytes	Normal	Accuracy	1→1
		Coverage	0.974→0.964
	ipsweep	Accuracy	1→1
		Coverage	0.17→0.16
	buffer_overflow	Accuracy	1→1
		Coverage	1→0.667
	warezmaster	Accuracy	1→0

		Coverage	1→0
	ftp_write	Accuracy	1→1
		Coverage	1→0.333
	multihop	Accuracy	1→0
		Coverage	1→0
	rootkit	Accuracy	1→0
		Coverage	1→0
land	land	Accuracy	1→1
		Coverage	1→0.25
wrong_fragment	teardrop	Accuracy	1→1
		Coverage	0.988→0.105
hot	warezclient	Accuracy	1→1
		Coverage	0.464→0.26
count	neptune	Accuracy	1→1
		Coverage	0.545→0.138
	satan	Accuracy	1→1
		Coverage	0.769→0.763
	smurf	Accuracy	1→1
		Coverage	0.735→0.723
srv_count	smurf	Accuracy	1→1
		Coverage	0.735→0.265
serror_rate	neptune	Accuracy	1→1
		Coverage	0.545→0.54
	satan	Accuracy	1→1
		Coverage	0.769→0.753

rerror_rate	satan	Accuracy	1→1
		Coverage	0.769→0.484
same_srv_rate	neptune	Accuracy	1→1
		Coverage	0.545→0.502
	satan	Accuracy	1→1
		Coverage	0.769→0.737
diff_srv_rate	satan	Accuracy	1→1
		Coverage	0.769→0.742
	portsweep	Accuracy	1→1
		Coverage	0.606→0.515
dst_host_count	normal	Accuracy	1→1
		Coverage	0.974→0.971
dst_host_srv_count	normal	Accuracy	1→1
		Coverage	0.974→0.97
dst_host_diff_srv_rate	nmap	Accuracy	1→1
		Coverage	0.186→0.14
dst_host_same_src_port_rate	normal	Accuracy	1→1
		Coverage	0.974→0.973
dst_host_srv_diff_host_rate	normal	Accuracy	1→1
		Coverage	0.974→0.972
	warezclient	Accuracy	1→1
		Coverage	0.464→0.453
	ipsweep	Accuracy	1→1
		Coverage	0.17→0.132
dst_host_serror_rate	satan	Accuracy	1→1

		Coverage	0.769→0.742
	nmap	Accuracy	1→1
		Coverage	0.186→0.07
dst_host_srv_serror_rate	guess_passwd	Accuracy	1→1
		Coverage	0.4→0.067
dst_host_rerror_rate	satan	Accuracy	1→1
		Coverage	0.769→0.763
	portsweep	Accuracy	1→1
		Coverage	0.606→0.591
dst_host_srv_rerror_rate	normal	Accuracy	1→1
		Coverage	0.974→0.96

而我們由表 11 也可以得知對於 normal 有影響的屬性有 8 個分別是{service,src_bytes,dst_bytes,dst_host_count,dst_host_srv_count,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_srv_rerror_rate}，因此在擴充版本一中的入侵偵測系統架構部份便使用 8 個屬性去偵測所有的資料量，然後判斷出"正常"及"異常"，接著再針對有興趣的決策類別使用其應該用的條件屬性來判斷"異常"部分的資料，因此可由原本需要 41 種屬性的龐大資料量，變成只使用 8 個屬性的去判斷是否屬於 normal 的決策類型，若不屬於 normal 則屬於攻擊類型，即"異常"部分，因此在資料量明顯降低的情況下，處理速度會比原本的快速許多。

經過兩階段屬性縮減後我們可以由規則分類所得到結果如表 12，由訓練集我們可以看到與原本使用 41 種條件屬性相比，使用 22 種屬性時在訓練集資料覆蓋率略降 0.01，而在測試集資料則是準確率降低 0.01，但是覆蓋率卻提升 0.01，我們可以推測這是由於覆蓋率提升而導致準確率下降，而在覆蓋率提升時準確率下降很小，因此這種結果是我們可以接受的。

表 12 ： 訓練集及測試集比較

資料集	條件屬性數目	分類器(rule-based)	
訓練集資料（兩萬筆）	22 個	Accuracy	1
		Coverage	0.863
	41 個	Accuracy	1
		Coverage	0.864
測試集資料（八萬筆）	22 個	Accuracy	0.989
		Coverage	0.861
	41 個	Accuracy	0.99
		Coverage	0.86

最後我們可以由圖 16 和圖 17 看到使用了 41 種條件屬性的資料集範本和使用 22 種條件屬性的資料集範本，資料量明顯減少了將近一半，而由圖 18 也可以看出，使用 8 種屬性資料集的資料量更是明顯少很多，因此我們縱向減少了大量不必要的訊息，相信入侵偵測系統所需要處理的資料量也會大幅減少，而若在只需要知道是不是攻擊之狀況下，即不需要知道攻擊類型更可直接使用判斷 normal 的 8 種屬性，將可大量提升入侵偵測系統的效能，在我們的兩階段屬性縮減的結果表中也能夠得知哪些攻擊類型需要哪些屬性，因此我們也能夠依造我們的需求去添加我們所需要的條件屬性，進而只找出有興趣的攻擊類型。

0 tcp http SF 232 2654 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 16 17 0 0 0 0 1 0 0.12 206 242 1 0 0 0.04 0 0 0 0
0 tcp http SF 288 1046 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 3 3 0 0 0 0 1 0 0 3 255 1 0 0.33 0.03 0 0 0 0
0 tcp http SF 345 6055 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 8 13 0.12 0.08 0 0 1 0 0.31 8 255 1 0 0.12 0.08 0.12 0.01 0 0
0 tcp http SF 164 2112 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 8 10 0 0 0 0 1 0 0.2 34 255 1 0 0.03 0.03 0 0 0 0
0 tcp http SF 273 4471 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 24 24 0 0 0 0 1 0 0 136 255 1 0 0.01 0.04 0 0 0 0
0 tcp smtp SF 3446 321 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 3 0 0 0 0 1 0 0.67 114 183 0.91 0.03 0.01 0.01 0 0 0 0
0 tcp http SF 297 770 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 5 42 0 0 0 0 1 0 0.05 60 255 1 0 0.02 0.04 0 0 0 0.01
0 tcp http SF 207 376 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 3 20 0 0 0 0 1 0 0.35 29 255 1 0 0.03 0.04 0 0 0 0
0 tcp private S0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 129 3 1 1 0 0 0.02 0.08 0 255 3 0.01 0.08 0 0 1 1 0 0
0 tcp http SF 287 1572 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 6 7 0 0 0 0 1 0 0.28 13 255 1 0 0.08 0.03 0 0 0 0

圖 16 ： 使用 41 種條件屬性的原始資料集前 10 筆範本

```
0 http 232 2654 0 0 0 16 17 0 0 1 0 206 242 0 0 0.04 0 0 0 0
0 http 288 1046 0 0 0 3 3 0 0 1 0 3 255 0 0.33 0.03 0 0 0 0
0 http 345 6055 0 0 0 8 13 0.12 0 1 0 8 255 0 0.12 0.08 0.12 0.01 0 0
0 http 164 2112 0 0 0 8 10 0 0 1 0 34 255 0 0.03 0.03 0 0 0 0
0 http 273 4471 0 0 0 24 24 0 0 1 0 136 255 0 0.01 0.04 0 0 0 0
0 smtp 3446 321 0 0 0 2 3 0 0 1 0 114 183 0.03 0.01 0.01 0 0 0 0
0 http 297 770 0 0 0 5 42 0 0 1 0 60 255 0 0.02 0.04 0 0 0 0.01
0 http 207 376 0 0 0 3 20 0 0 1 0 29 255 0 0.03 0.04 0 0 0 0
0 private 0 0 0 0 0 129 3 1 0 0.02 0.08 255 3 0.08 0 0 1 1 0 0
0 http 287 1572 0 0 0 6 7 0 0 1 0 13 255 0 0.08 0.03 0 0 0 0
```

圖 17 ： 使用 22 種條件屬性的原始資料集前 10 筆範本

```
http 232 2654 206 242 0 0.04 0
http 288 1046 3 255 0.33 0.03 0
http 345 6055 8 255 0.12 0.08 0
http 164 2112 34 255 0.03 0.03 0
http 273 4471 136 255 0.01 0.04 0
smtp 3446 321 114 183 0.01 0.01 0
http 297 770 60 255 0.02 0.04 0.01
http 207 376 29 255 0.03 0.04 0
private 0 0 255 3 0 0 0
http 287 1572 13 255 0.08 0.03 0
```

圖 18 ： 使用 8 種條件屬性的原始資料集前 10 筆範本

4.3 IDS 分類方法比較

接下來我們分別使用不同的 IDS 常用分類器去進行實驗比較，刪減屬性前後是否會影響入侵偵測系統分類上的準確率和覆蓋率。首先我們使用約略集研究系統提供的 Decomposition Tree[23]，由表 13 可以看出使用我們分別使用訓練集資料去看縮減前準確率為 1，覆蓋率為 0.997，縮減後準確率仍為 1，覆蓋率也為 0.997，而在測試集資料準確率的變化也是 1→1，覆蓋率的變化 0.998→0.998，因此不論在訓練集資料和測試集資料的準確率和覆蓋率完全沒有降低，仍相當高。

表 13 ： 使用 Decomposition Tree 分類方法比較屬性縮減影響

資料集	條件屬性數目	分類方法(Decomposition Tree)	
訓練集資料（兩萬筆）	22 個	Accuracy	1
		Coverage	0.997
	41 個	Accuracy	1
		Coverage	0.997
測試集資料（八萬筆）	22 個	Accuracy	1
		Coverage	0.998
	41 個	Accuracy	1
		Coverage	0.998

表 14 ： 使用 NN 分類方法比較屬性縮減影響

資料集	條件屬性數目	分類方法(NN)	
訓練集資料（兩萬筆）	22 個	Accuracy	0.992
		Coverage	1
	41 個	Accuracy	0.995
		Coverage	1
測試集資料（八萬筆）	22 個	Accuracy	0.996
		Coverage	1
	41 個	Accuracy	0.996
		Coverage	1

接著我們使用類神經網路(NN)[24]來當分類器進行研究，由表 14 能夠看出使用訓練集資料使用 41 個條件屬性覆蓋率為 1，準確率為 0.995，而使用 22 個條件屬性時，準確率沒變，仍為 1，而準確率從 0.995 降為 0.992，但是我們可以看再

更大量的測試集資料下，覆蓋率縮減前和縮減後一樣仍是 1，而準確率也是沒有改變縮減前和縮減後都是 0.996，因此實驗縮減移除後剩餘的 22 個條件屬性對於類神經網路的分類上也能維持一定的覆蓋率及準確率。

表 15 ： 使用 KNN 分類方法比較屬性縮減影響

資料集	條件屬性數目	分類方法(KNN)	
訓練集資料（兩萬筆）	22 個	Accuracy	1
		Coverage	1
	41 個	Accuracy	1
		Coverage	1
測試集資料（八萬筆）	22 個	Accuracy	1
		Coverage	1
	41 個	Accuracy	1
		Coverage	1

最後我們使用 K 個最近鄰居法(KNN) [25][26]來當分類器的實驗，由表 15 可以看出 KNN 的分類法更能夠明顯的表示出我們不論使用訓練集資料或測試集資料，屬性縮減前和縮減後的準確率和覆蓋率都十分的高，均為 1。

因此由上述之三種實驗可知，經由 RST 二階段屬性縮減分析後的 22 個所需屬性為一可信及可用的實驗結果。

4.4 屬性縮減相關研究比較

本節將根據我們在相關文獻所介紹不同的入侵偵測機制在判斷攻擊時所使用的不同的 KDD-Cup 1999 資料集屬性數[5][7]來分析比較。H.Hom 等人他們提出

不使用離散的和符號型態的屬性來進行實驗，因此我們根據 KDD-Cup 1999 資料集所提供的型態區分，可以看出{protocol_type, service, flag, land, logged_in, is_host_login, is guest_login}這七個屬性都是符號型態而其餘屬性都是連續型態的，因此我們實驗將移除這七個屬性去進行實驗，實驗結果我們可以由圖 19 和圖 20 看出 normal、neptune、pod 和 land 都會受到影響，normal 覆蓋率由 0.974 降為 0.973，neptune 覆蓋率由 0.545 降為 0.512，pod 覆蓋率由 1 降為 0.958，land 覆蓋率由 1 降為 0.25，因此我們可以看出 H.Hom 等人刪除的屬性數不多，所使用的屬性遠遠多於我們所使用的屬性數目，而且 H.Hom 等人刪除到許多重要的屬性導致對於分類攻擊會有很大的影響，使得 normal、neptune、pod 和 land 的覆蓋率都降低，而我們的方法雖然刪除的屬性數量比 H.Hom 等人提出的還多，但是經過實驗證實我們所刪除的屬性對於分類並無影響，因此準確率及覆蓋率仍能夠保持水準。由上述我們可以得知對於分類要得到最好的效果，並不需要使用過多的屬性數目，只需要使用有助於分類的屬性。

	normal.	neptun...	satan.	teardr...	back.	warez...	ports...	ipswe...	smurf.	buffer...	guess...	nmap.	warez...	pod.	ftp_w...	land.	imap.	multi...	rootkit.	No. of obj.	Accuracy	Coverage
normal.	14,169	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14,545	1	0.974
neptune.	0	2,436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4,468	1	0.545
satan.	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	1	0.769
teardrop.	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	1	0.988
back.	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	1
warezclient.	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0.464
portsweep.	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	66	1	0.606
ipsweep.	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	106	1	0.17
smurf.	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	83	1	0.735
buffer_overflow.	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	6	1	1
guess_pass...	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	15	1	0.4
nmap.	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	43	1	0.186
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	6	1	1
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	24	1	1
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	1	1
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	4	1	1
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	1	1
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

圖 19 ： 使用本研究 22 個屬性所得結果

	normal.	neptune.	satan.	teardr...	back.	warezc...	ports...	ipswe...	smurf.	buffer...	guess...	nmap.	warezm...	pod.	ftp_w...	land.	imap.	multi...	rootkit.	No. of obj.	Accuracy	Coverage
normal.	14,151	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14,545	1	0.973
neptune.	0	2,288	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4,468	1	0.512
satan.	0	0	143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	186	1	0.769
teardrop.	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	1	0.988
back.	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173	1	1
warezclient.	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0	0	0	0	0	181	1	0.464
portsweep.	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	66	1	0.606
ipsweep.	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	106	1	0.17
smurf.	0	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	83	1	0.735
buffer_overflow.	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	6	1	1
guess_pass...	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	15	1	0.4
nmap.	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	43	1	0.186
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	6	1	1
pod.	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	24	1	0.958
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	3	1	1
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4	1	0.25
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	1	1
multihop.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

圖 20 ： 使用 H. Hom 所刪減後的 34 屬性的分類結果

而 S.Mukkamala 提出使用 20 種第一重要的屬性{duration,service,src_bytes, dst_bytes,wrong_fragment,num_access_files,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,dst_host_count, dst_host_srv_count, dst_host_diff_srv_rate, dst_host_same_src_port_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate}，以及加上第二重要的屬性{protocol_type,land, urgent,num_failed_logins,root_shell,num_file_creations,num_outbound_cmds,is_guest_login,same_srv_rate,diff_srv_rate,dst_host_same_srv_rate,dst_host_srv_diff_host_rate }，因此我們分別進行實驗。

經由我們整理過後可以由表16看出使用20個屬性和使用我們的22個屬性對於 Decomposition Tree來說是沒有影響的。可是我們再使用類神經網路(NN)當做分類器經由整理後可以得到表17，可以看到訓練集使用20個屬性的覆蓋率為1準確率為0.986和使用22個屬性覆蓋率1為準確率為0.992，再測試資料集更加明顯，準確率分別為0.987及0.996，因此我們使用22種屬性在類神經網路分類方法中是優於S.Mukkamala所提出的20種屬性。最後我們也使用K個最近鄰居法(KNN)進行實驗，可以由表18看到效果仍然很好，但是由我們的實驗結果圖21可以看出使用20種屬性會使得pod的準確率由1降為0.979，雖說影響不大，但是入侵偵測的用途就

是不放過任何攻擊，因此若使用我們提出的22種屬性在KNN分類方法下將不會導致任何攻擊的準確率下降，所以經由類神經網路和K個最近鄰居比較，可以看出我們使用22個屬性所獲得的效果比較好。接著我們使用他提出的20個屬性加上第二重要的12個屬性一共32個屬性分別以Decomposition Tree和類神經網路為分類器的實驗，從表16可以看出在使用Decomposition Tree分類方法下，32個屬性和22個屬性並無差別，而從表17可以看出使用類神經網路分類方法下，雖然在訓練集資料下使用22個屬性比32個屬性的準確率少0.01，但是在測試集資料下卻是同樣為0.996的準確率，因此在經過多次實驗過後，我們認為使用22種屬性確實能夠比使用20種屬性好，而且不必使用到32個屬性，22個屬性便已經足夠。

表 16 ： 使用不同條件屬性於 Decomposition Tree 下之比較

資料集	條件屬性數目	分類方法(Decomposition Tree)	
訓練集資料（兩萬筆）	20 個	Accuracy	1
		Coverage	0.997
	22 個	Accuracy	1
		Coverage	0.997
	32 個	Accuracy	1
		Coverage	0.997
測試集資料（八萬筆）	20 個	Accuracy	1
		Coverage	0.998
	22 個	Accuracy	1
		Coverage	0.998
	32 個	Accuracy	1
		Coverage	0.998

表 17 ： 使用不同條件屬性於 NN 下之比較

資料集	條件屬性數目	分類方法(NN)	
訓練集資料（兩萬筆）	20 個	Accuracy	0.986
		Coverage	1
	22 個	Accuracy	0.992
		Coverage	1
	32 個	Accuracy	0.993
		Coverage	1
測試集資料（八萬筆）	20 個	Accuracy	0.987
		Coverage	1
	22 個	Accuracy	0.996
		Coverage	1
	32 個	Accuracy	0.996
		Coverage	1

表 18 ： 使用不同條件屬性於 KNN 下之比較

資料集	條件屬性數目	分類方法(KNN)	
訓練集資料（兩萬筆）	20 個	Accuracy	1
		Coverage	1
	22 個	Accuracy	1
		Coverage	1
測試集資料（八萬筆）	20 個	Accuracy	1
		Coverage	1
	22 個	Accuracy	1
		Coverage	1

	normal.	neptu...	gues...	back.	nmap.	ports...	teardr...	satan.	pod.	multih...	ipswe...	smurf.	warezc...	loadmod...	root...	buffer...	land.	warez...	imap.	spy.	ftp_w...	No. of obj.	Accuracy	Coverage
normal.	29,126	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29,126	1	1
neptune.	0	8,944	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8,944	1	1
guess_passw..	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	1	1
back.	0	0	0	350	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	350	1	1
nmap.	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	61	1	1
portsweep.	0	0	0	0	0	133	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	133	1	1
teardrop.	0	0	0	0	0	0	223	0	0	0	0	0	0	0	0	0	0	0	0	0	0	223	1	1
satan.	0	0	0	0	0	0	0	372	0	0	0	0	0	0	0	0	0	0	0	0	0	372	1	1
pod.	1	0	0	0	0	0	0	0	47	0	0	0	0	0	0	0	0	0	0	0	0	48	0.979	1
multihop.	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	5	1	1
ipsweep.	0	0	0	0	0	0	0	0	0	0	223	0	0	0	0	0	0	0	0	0	0	223	1	1
smurf.	0	0	0	0	0	0	0	0	0	0	0	130	0	0	0	0	0	0	0	0	0	130	1	1
warezclient.	0	0	0	0	0	0	0	0	0	0	0	0	346	0	0	0	0	0	0	0	0	346	1	1
loadmodule.	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	5	1	1
rootkit.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	3	1	1
buffer_overflow.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	1	1
land.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	4	1	1
warezmaster.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	5	1	1
imap.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	1	1
spy.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
ftp_write.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

圖 21 ： 使用 20 個屬性於 KNN 分類方法的實驗結果

五、 結論與未來工作

本研究是以藉由降低屬性數目間接減少入侵偵測系統所需處理的資料量，進而達到在維持入侵偵測系統的偵測準確率不變下，提升入侵偵測系統之效能。

從實驗結果可以看出，我們使用我們提出的兩種方法從訊息系統刪除沒幫助的屬性訊息處理過後，所需的屬性從 41 種大幅降低成為 22 種，可以由表 12 看出準確率在訓練資料集沒有下降，但在測試資料集的比較中刪除某些屬性會造成某些類別判斷的準確率略微上升或下降，但是所造成的效果平均過後並不會差太多，而和其他研究所使用的屬性數目相較之下也明顯下降，證明我們的方法是可行及有效率的。

我們也使用 Decomposition Tree、類神經網路(NN)和 K 個最近鄰居法來進行較多的比較，實驗結果也證明使用縮減後的屬性，還是能夠維持縮減前的準確率及覆蓋率。最後我們再與其他研究中所建議的屬性數目來做比較，由實驗結果可知本文所建議的屬性數目的確比其他研究使用的少，雖然 S.Mukkamal 提出可以使用 20 種屬性，但我們也經由實驗證明使用類神經網路時，所提出的 22 種屬性的準確率比起使用 20 種好，而在 K 個最近鄰居也能夠使 pod 的準確率達到最高，因此使用 22 種屬性對其它分類器的確能夠獲得比較好的成果。

雖然我們用以支援入侵偵測系統判別攻擊的屬性建議為 22 個，但是或許還有降低的空間，也有可能我們刪除的屬性特徵對於某些分類器有助其得到更好的準確率，因此未來可再配合使用約略集理論以外的方法再去減少屬性數目，希望能夠挑選出最精簡且最正確的屬性特集合。且於本研究中我們在 IDS 分類核心是使用 Decomposition Tree、類神經網路(NN)和 K 個最近鄰居法來進行實驗，或許可

以再探討本研究的結果若在 IDS 分類核心中使用 SVM 及 HMM(hidden Markov model)等其他 IDS 分類核心的效能，測試是否使用 22 個屬性會影響 IDS 的分類準確率。

最後針對使用 8 種屬性對於 normal 判斷的架構，此種方法雖然能夠有效的分類出正常及異常類型的資料，也能夠針對未知的攻擊手法來進行分類，但是若對於 normal 的準確率及覆蓋率沒有達到百分百，可能會造成入侵偵測系統誤判過多的正常資料為攻擊資料，因此未來可以增加對 normal 的覆蓋率，使其判斷為正常的資料增加可信度，進而由不屬於正常資料即為異常資料，更能夠應用於判斷更多未知的攻擊手法。

參考文獻:

[1] Dr. Dirk Ourston, Ms. Sara Matzner, Mr. William Stump, and Dr. Bryan Hopkins, ”Applications of Hidden Markov Models to Detecting Multi-stage Network Attacks”, 6-9 Jan. 2003, On page(s): 10 pp.

[2] 曾憲雄、蔡秀滿、蘇東興、曾秋蓉、王慶堯(2006)， 資料探勘 , Data Mining , 2006 , 3 月

[3] KDD Cup Data set , http://www.sigkdd.org/kddcup/index.php

[4] Sandhya Pedabchiguri, Ajith Abrahm, Crina Grosan, Johnson Thomas, “Modeling intrusion detection system using hybrid intelligent systems.”, Journal of Network and Computer application June 2005, On page(s): 114–132, 2007

[5] Hung Hom, Kowloon, “DDos Detection based on feature space Modeling., Machine Learning and Cybernetics”, 2004. Proceedings of 2004 International Conference, Aug. 2004 , On page(s): 4210- 4215 vol.7

[6] S. Mukkamala, A H. Sung, “Identifying Key Features for Intrusion Detection Using Neural Networks.“ Proceedings of 15th International Conference on Computer Communications, On page(s): 1132-1138

[7] Mukkamala, S. Sung, A.H. (2003), “Detecting denial of service attacks using support vector machines”, Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference, On page(s): 1231- 1236 vol.2

[8] N. Zhang and W. F. Lu, “An Efficient Data Preprocessing Method for Mining Customer Survey Data”, Industrial Informatics , IEEE International Conference , On Page(s): 573-578**,** 2007 5th

[9] H. Mirghasemi , M. B. Shamsollahi, and R. Fazel-Rezai, “Assessment of Preprocessing on Classifiers Used in the P300 Speller Paradigm.”, Conf Proc IEEE Eng Med Biol Soc. April 24, 2006., On Page(s):1319-1322

[10] Ping-Feng Pai, Wan-Ru Wei, “Predicting Movement Directions of Stock Index Futures by Support Vector Models with Data Preprocessing.”, Industrial Engineering and Engineering Management , 2007 IEEE International Conference , On page(s): 169-173

[11] H.Hannah Inbarani, K.Thangavel, A. Pethalakshmi, “Rough set based Feature Selection for Web Usage Mining.”, Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference , Dec. 2007 , On page(s): 33-38

[12] Pawlak, Z., “Rough Sets and Intelligent Data Analysis”, Information Sciences , Vol. 147, 2002, pp. 1-12.

[13] B. Walczak , D.L. Massart , “Tutorial Rough sets theory” , Chemometrics and Intelligent Laboratory Systems 47 (1999) 1-16

[14] 齊立文 ,“80-20 法則 聰明工作，認真玩”, 經理人月刊, 第三十八期, 2008.01.01

[15] Rough Set Exploration System(RSES) , http://logic.mimuw.edu.pl/~rses/

[16] 賴威利 ,“利用約略集理論預測燒燙傷患者死亡率”, 全國碩博士論文網 , 2005.7

[17] 陳士杰 ,“應用資料探勘技術於理財促銷-以國內某金控銀行為例”, 全國碩博士論文網 , 2005

[18] 施奕良 ,“知識表達方法於影像判釋之研究－以粗糙集合理論與主成分分析為例”, 全國碩博士論文網 , 2006.6

[19] J. Bazan, M. Szczuka, “RSES and RSESlib - A Collection of Tools for Rough Set Computations (Postscript)”. Extended version of paper presented at RSCTC'2000

[20] G. Bazan, Son H. Nguyen, Trung T. Nguyen, A. Skowron and J. Stepaniuk (1998). “Decision rules synthesis for object classification.“, In: E. Orowska (ed.),

Incomplete Information: Rough Set Analysis, Physica - Verlag, Heidelberg, pp. 23-57.

[21] J.G. Bazan. A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Table. In: L. Polkowski, A. Skowron (eds.),"Rough Sets in Knowledge Discovery", Physica - Verlag, Heidelberg, pp. 321-365.

[22] 許建隆,"《資訊安全》 網路駭客攻擊－分散式阻斷服務（DDoS）攻擊"：http://www.hurricanetech.net/

[23] Hoa S. Nguyen. "Data regularity analysis and applications in data mining.", Ph. D. thesis, supervisor B. Chlebus, Warsaw University.

[24] M. Wojnarski, LTF-C: Architecture, "Training Algorithm and Applications of New Neural Classifier." Fundamenta Informaticae, Vol. 54(1), pp. 89–105. IOS Press, 2003

[25] G. Gora, A. Wojna, "RIONA: A Classifier Combining Rule Induction and k-NN Method with Automated Selection of Optimal Neighbourhood", Proceedings of the Thirteenth European Conference on MachineLearning, ECML 2002, Helsinki, Finland, Lecture Notes in Artificial Intelligence, 2430, Springer-Verlag, pp. 111–123

[26] G. Gora, A. Wojna, "RIONA: A New Classification System Combining Rule Induction and Instance-Based Learning", Fundamenta Informaticae, 51(4), pp. 369–390