國立交通大學

資訊學院資訊科技（IT）產業研發碩士專班

資料庫軟體上的資訊安全
Information Security on Warehouse Application

研 究 生：巫祈賢

指導教授：曾文貴　教授

中華民國壹百年陸月

資料庫軟體上的資訊安全

Information Security on Warehouse Application

研 究 生：巫祈賢　　Student：Chi-Hsien Wu

指導教授：曾文貴　　Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資訊學院資訊科技（IT）產業研發碩士專班

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on

Computer Science and Engineering

June 2011

Hsinchu, Taiwan, Republic of China

中華民國壹百年陸月

資料庫軟體上的資訊安全

學生：巫祈賢　　指導教授：曾文貴 教授

國立交通大學資訊學院產業研發碩士專班

摘要

隨著網際網路的蓬勃發展及越來越多的資料庫服務系統，個人的情資變得更容易被有心人士取得，其中除了外部的資料竊取外，比較嚴重的是內部人員利用職務之便的資料外洩事件。因此在此論文中，我們利用機器學習中的類神經網路及統計分析技術等，在內部人員輸入的資料庫查詢指令送到資料庫前，甚至是資料庫管理者在執行資料庫指令時，我們便迅速記錄這些資料庫指令的操作，使得之後要追蹤這些資料庫指令，是由資料庫管理者或哪些內部人員所下達時，能更迅速找到源頭，而目前的網路服務的平台大多採用三層式的服務架構。

在本篇論文中，我們在盡量不變更三層式網路服務架構下，設計了一套資料庫稽查方法，然後我們利用 http 封包與 SQL 封包的相關特性及類神經網路來完成我們的稽查演算法。我們設計的方法能在不知道第二層 AP 伺服器的架構下達到 55%的稽查準確度，而且達到每秒 450 筆 SQL 指令稽查的效果。此外，經由這樣的資料庫指令稽查，日後我們能容易地找出資料庫管理者更動了那些資料、以及其是否假公濟私盜竊了客戶個資等等，而這樣的情事都將在系統中受到監控，而變得無所遁形。

Information Security on Warehouse Application

Student : Chi-Hsien Wu Advisor : Dr. Wen-Guey Tzeng

Industrial Technology R & D Master Program of
Computer Science College
National Chiao Tung University

Abstract

When network becomes more and more popular, people disclose their privacy information more easily. Besides exterior attacks cause privacy information disclosure, interior threat is more serious. For example, interior members may have the right to access privacy information of other members in database legally. In this paper, we make use of neural network and statistical analysis method to achieve privacy-protecting goal. We record each database operation when somebody wants to access any information in the database. We analyze the operation, trace who sent it to the database, and then find out the operation sender. Nowadays many network service platforms take three layers architecture.

In the thesis, we design a database auditing system that would not change the three layers architecture too much. Besides, we use the relationship between http packets and SQL packets, and neural network to design our auditing algorithm. Our system could achieve 55 percent auditing accuracy and 450 SQL operation mapping without knowing the second AP server architecture first. In this way, we get effective database auditing and prevent privacy information from disclosing.

致謝

本篇論文能順利完成，最要感謝的便是我的指導教授曾文貴老師。曾老師在我研究的過程中，總是給我不同的建議，指導我方向、指導我做研究的方法、指導我在遇到問題時如何找方法去解決它們，這些不但使我能順利的完成我的論文，甚至也影響我日後遇到問題，懂得用抽絲剝繭及循序漸進的方式，一步步地找出問題的脈絡，然後切成小問題並加以解決，感謝老師。另外也感謝蔡錫鈞教授、謝續平教授及孫宏民教授擔任我的口試委員，在口試過程中提供了不少寶貴的想法及豐富的建議，讓本篇論文能更加完備，在此特別感謝。此外，謝謝實驗室的學長姐、同學們、及可愛的學弟妹。學長姐們的經驗分享，使我在做研究時有更寬廣的角度來思考問題；同學們的互相合作與幫忙，也讓我在遇到問題時能更迅速的找到較佳的解法；學弟妹的打氣與加油，更是讓我在挑燈夜戰地做著研究時，添增一份溫馨與關懷。總之，實驗室的大夥們，謝謝你們！最後我要感謝我的家人與朋友，謝謝你們一路走來的支持，讓我能沒有後顧之憂的完成碩士學業。

目錄

表目錄

圖目錄

第一章 緒論

隨著網際網路的蓬勃發展及越來越多的資料庫服務系統，個人的情資變得更容易被有心人士取得，其中除了外部的資料竊取外，比較嚴重的是內部人員利用職務之便的資料外洩事件。因此在此論文中，我們利用資料探勘中的類神經網路及統計分析技術等，在查詢的指令送入資料庫前，甚至是資料庫管理者在執行他指令時，我們便迅速記錄它們的動作，使得之後要追蹤那些資料庫指令是由那些人員所下達，甚至是資料庫管理者更動了那些資料、資料庫管理者是否假公濟私盜竊了客戶個資等等，都將在系統中受到監控，而變得無所遁形。

1.1 動機與目的

在現今的社會，由於網路的發達與資訊的蓬勃發展，大部分的民眾生活幾乎與網路上的服務息息相關，舉凡網路銀行、網路報稅、網路購物、網路搜索引擎等等(ex: google、yahoo)，而目前的網路服務的平台架構大多採用三層式的，如圖 1 所示。亦即在使用者在網路上要求服務時，會先連到一個使用者介面的伺服器，簡稱 UI 伺服器，之後再由此伺服器根據使用者所提出的服務要求，把該要求交給負責該服務的第二層的應用伺服器，簡稱 AP 伺服器(如圖 1 中的郵件伺服器、帳號管理伺服器、個人資料查詢伺服器等)，然後若是該服務有牽涉到資料庫應用的，再由 AP 伺服器依據該要求，而向資料庫採取一些相對應的動作，例如新增資料、刪除資料、查詢資料等等。

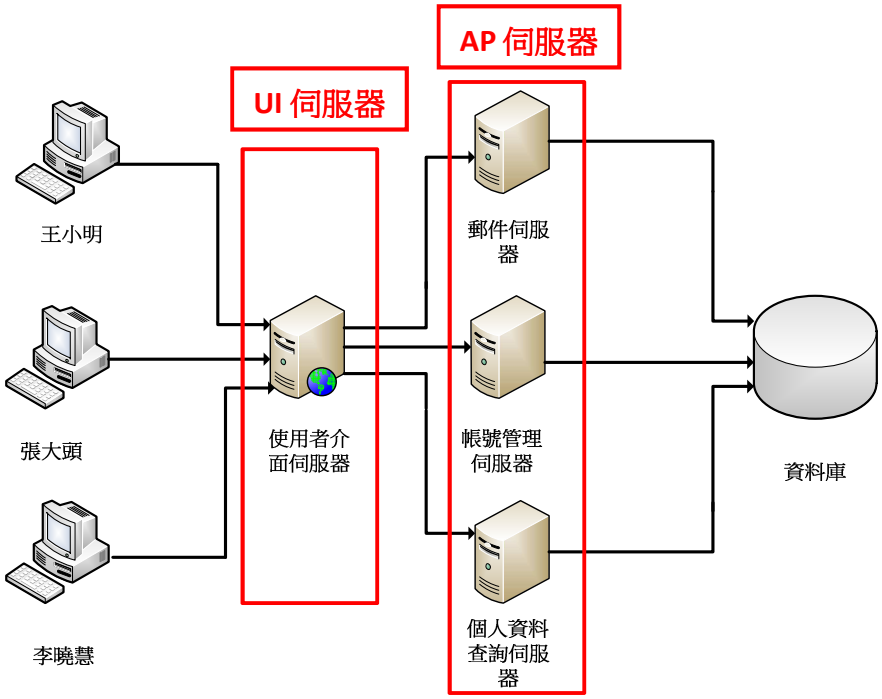


圖 1 三層式的網路服務架構

然而在這樣三層式的網路服務架構下，資料庫端的管理者為了能有效處理資料庫的問題，因此他們對於資料庫的操作常常擁有最高權限，而這也造成如果資料庫管理者私自更動資料庫內容的話，我們是很難發現的，更甚者如果資料庫權限沒有設定好，使得除了資料庫管理者外，也有其他人士可以對資料庫進行超出其權限的操作，例如公司內部非會計或監察人員卻可以查詢公司內部其他人薪資的資訊。而這樣的問題，對於服務大眾為主要業務的公司尤為重要，因為它們的資料庫內存的不只是公司的資料，而是成千上萬使用者的隱私，若是這樣的資訊外洩，那麼影響的範圍便會非常的龐大。如 2004 年的遠傳電信及 2007 年的中華電信使用者資訊外洩案，及很多電視購物或是網路購物平台也不時傳出資訊外洩的問題。

目前，越來越多的資料庫廠商看到了這個問題，也提出一些資料庫稽查的方法，讓資料庫發生資料外洩或是不正常的資料變更時，稽查人員可以迅速發現發生問題的資料庫指令是由哪些 AP 伺服器發出，進而循序找出洩密的源頭或是異常資料變更的來源。

而要解決這樣的問題，最快的方法就是在 UI 伺服器對 AP 伺服器送出要求服務的指令，以及 AP 伺服器對資料庫送出資料庫指令時，對每個資料庫指令加上一個序號，使後續追蹤能較為簡單，然而資料庫廠商所負責的部分往往只局限於資料庫的部分，至於 UI 伺服器與 AP 伺服器的設計就算不是由公司、政府資訊部門所設計，公司、政府資訊部門也會外包給其他廠商，因此在不清楚 UI 伺服器與 AP 伺服器架構下，要找出有問題的資料庫指令是由哪些 AP 伺服器所發出，進而找出問題的來源，便顯得相當困難了。

因此，為了解決上述的難題，我們在與資料庫廠商--庫柏資訊協商後，決定利用在他們的系統運作下所得到的資訊，來找出所有資料庫指令的來源，使得日後稽查人員在稽查時能迅速找出有問題的資料庫指令來源，以及避免資料庫管理者的監守自盜或是內部資料外洩的問題。

1.2 論文架構

本論文的整體架構如下，第二章描述的是資訊庫稽查的背景知識，及現今一些廠商他們的在此領域的研究發展。第三章談的是與本文相關的一些相關研究，說明目前資料探勘的技術及決策樹、類神經網路等等機器學習的方法。第四章講的是我們的系統設計，將我們資料庫稽查系統分成學習及回憶兩個步驟，及詳細描述每個步驟的實作流程。第五章則是我們在做完實驗後所獲得的一些實驗結果探討與分析。而最後一章則是我們對於此問題做個總結及探討未來的方向。

第二章 背景知識

在本章節將介紹過去到現在發生過的一些資訊外洩事件及資料庫稽查的重要性，及目前一些目前資料庫廠商他們對於解決此問題的系統介紹。2.1 介紹一些與資料庫稽查相關的法案。2.2 探討目前資料庫廠商的一些解決方案。2.3 介紹資料庫廠商 Cobrasonic 解決此問題所用到的系統架構及我們的研究環境。

2.1 相關法案回顧

在 2002 年的美國發生安隆、世界通訊一連串公司財務報表不實的事件後，美國政府便訂定沙賓法案，該法案中很重要的一點就是要公司制定誠實、可靠的財務報表，然而這些財務報表的原始資料往往儲存在公司內部的資料庫裡，因此如何有效的稽查資料庫操作，以確保資料庫內資料的正確性及不會被蓄意的修改，便成了日後稽核人員最關心的議題。

除此之外，基於國內個人資料外洩問題嚴重，因此政府也擬定個資法，來防治個人資料外洩的問題。本法案將對外洩者或外洩公司採取一罪一罰的方法，以立法的方式杜絕個資外洩問題。而其中影響最大者便是以服務業為主的公司(ex：電信業者、金融業者)，因為這些公司最常接觸大量使用者的資料，而且一旦發生資料外洩，影響的往往是成千上萬筆的個資。除此之外，這樣龐大數量的資料外洩事件往往是內部員工所為，因此如何有效的稽查資料庫操作，使事先能預防個資外洩事件，或是在發生外洩事件後能迅速找出外洩來源，便成了一件非常重要的事了。

2.2 資料庫稽查的一些方案

而國內外許多資料庫廠商(ex: Oracle、Sentrigo、Imperva、Cobrasonic, etc.)針對這樣的問題開發了一些資料庫稽查系統，且各有其優缺點。像 Sentrigo 開發的 Hedgehog 系統，該系統在資料庫端有內建小型資料庫專門儲存資料庫操作的紀錄檔及後續追查的功能，然而由於儲存紀錄檔的小型資料庫容量有限，因此若是用在大量資料庫操作的環境下，很容易超出小型資料庫的容量。另外與之搭配的 AP 伺服器也必須限制在 IBM WebSphere、BEA WebLogic、Apache Tomcat、JBoss 等，或者 Microsoft .NET 的 IIS，若是舊版的 ASP 程式，便無法判讀。Hedgehog 的截圖如圖 2 所示。

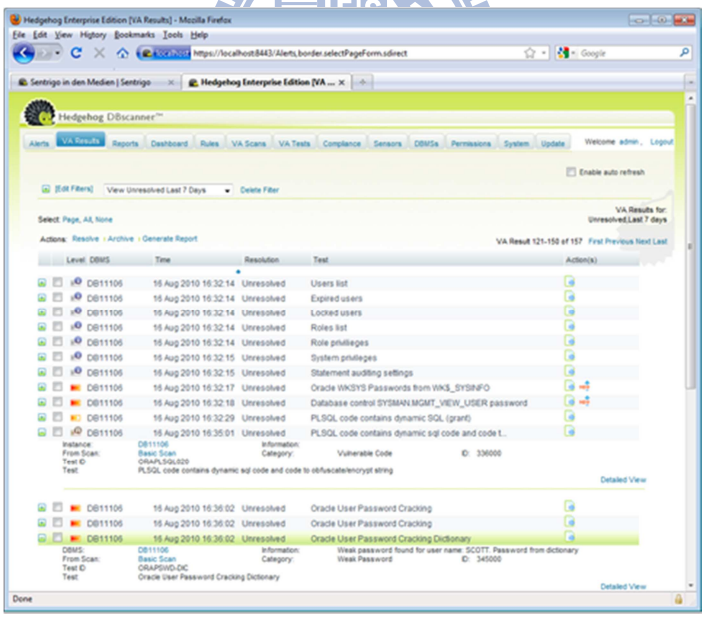


圖 2 Hedgehog 的使用介面

而 Imperva 所開發的 SecureSphere 則是把三層式網路服務架構濃縮成兩層的網路服務架構，然後把 UI 伺服器及 AP 伺服器結合成他們定義的 SAPFinance1 伺服器，這樣在整體架構都是 Imperva 設計的情況下，能達到更精準的資料庫稽查效果，然而要達到這樣的效果是採用 SecureSphere 系統的公司必須在 Imperva 的 SAPFinance1 伺服器上建構原本 UI 伺服器及 AP 伺服器的功能，不然就沒辦法達到這樣的效果了。SecureSphere 的架構如圖 3 所示。

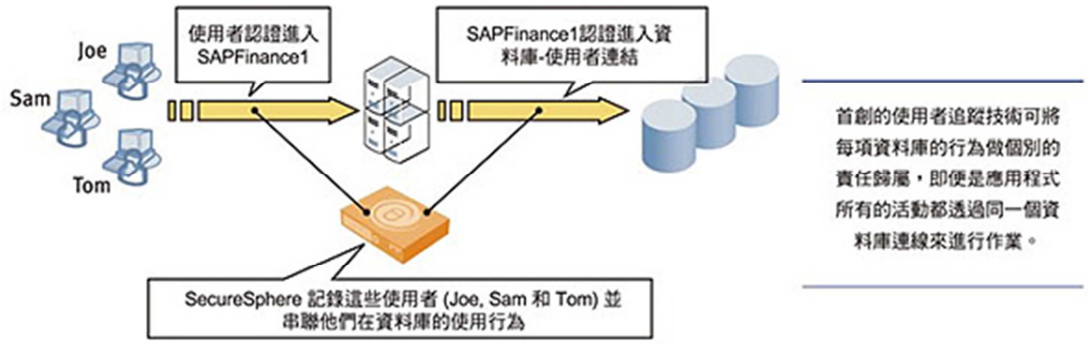


圖 3 SecureSphere 架構圖

2.3 Cobrasonic 的資料庫稽查系統架構

而 Cobrasonic 所設計的系統則是比較貼近三層式伺服器的架構，而且他們是在盡量不影響原本三層式伺服器的架構下，採取 UI 伺服器及資料庫封包拷貝後送到第三方伺服器，以進行後續分析的方式，如圖 4 所示。

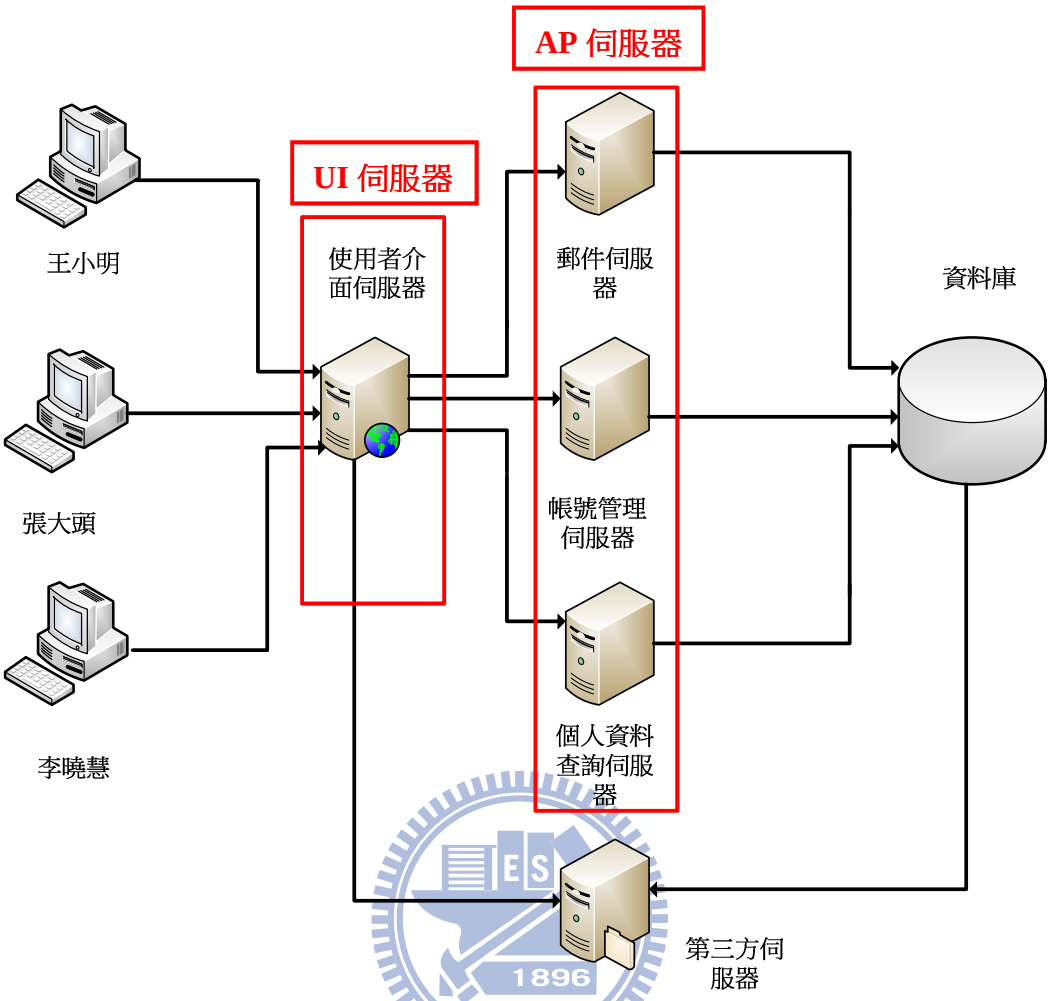


圖 4 Cobrasonic 資料庫稽查系統的架構

在 Cobrasonic 所設計的架構中，是在 UI 伺服器傳送 http 封包給 AP 伺服器時順道拷貝一份然後傳送給第三方伺服器，然後在資料庫收到 AP 伺服器傳來的 SQL 指令或是資料庫管理者執行資料庫操作動作時，也拷貝一份 SQL 指令給第三方伺服器，之後在第三方伺服器上分析 SQL 指令與 http 封包的關聯性，以達到對原系統架構影響最小卻又能達到資料庫稽查的效果。由於使用者在要求服務過程中的 http 封包及之後資料庫上操作的 SQL 指令一開始就會送一份給第三方伺服器，而且所有資料庫管理者在操作資料庫時也會同時把操作資訊傳送一份到第三方伺服器，因此之後發生問題時，稽核人員便可以很迅速的找到問題的來源，並加以解決了。

然而在這樣的設計架構下，最難解決的就是如何在不知道 AP 伺服器的架構下，利用 UI 伺服器送給它的 http 封包及 AP 伺服器送給資料庫的 SQL 指令中找出關聯性，進而能回推該 SQL 指令與哪筆 http 封包較有關係及該 SQL 指令是從那個 AP 伺服器所送出的。

以 Cobrasonic 的系統架構為例，在 UI 伺服器送往 AP 伺服器間的 http 封包格式，及 AP 伺服器送往資料庫之間包含 SQL 指令的封包格式分別如表 1 與表 2 所示。

時間郵戳	AP 伺服器中的網頁 URI	AP 伺服器中的網頁標籤名稱	AP 伺服器中的網頁標籤數值

表 1 http 封包格式

時間郵戳	SQL 指令

表 2 包含 SQL 指令的封包格式

假設今天上述三層式的網路架構是為了建構網路銀行所設計，那麼當使用者輸入帳號密碼要登入系統時，UI 伺服器會先發送 http 封包給 AP 伺服器，如表 3 所示，

時間郵戳	AP 伺服器中的網頁 URI	AP 伺服器中的網頁標籤名稱	AP 伺服器中的網頁標籤數值
1279180201	/aiad/pack01/ec/ec-login.asp	pp	6
1279180201	/aiad/pack01/ec/in/pos-pass-check.asp	passwd	1234567

表 3 發送帳號密碼的 http 封包

然後 AP 伺服器在收到 http 封包時，會送出 SQL 封包給資料庫，如表 4 所示。

時間郵戳	SQL Statement
1279180203	select userInfo from User where Password = “1234567”

表 4 與發送登入系統的 http 封包相對應的 SQL 封包

由上面的例子，我們可以看到 UI 伺服器送出的 http 封包內含有標籤名稱為”passwd”及標籤數值為”1234567”的內容，而且後續對應的 SQL 封包內容也含有數值為”1234567”的 Password 欄位，因此，我們很容易就可以發現它們是極有相關性的。

這是個比較好比對的情形，然而有些 http 封包內，紀錄 passwd 的欄位可能取名為 loginkey 或其他標籤名稱，而且同時間含有相同標籤數值的 http 封包可能超過一筆，在這樣的情況下，簡單的標籤名稱及標籤數值比對，便無法找到相對應的 SQL 封包與 http 封包了，而這也是 Cobrasonic 所設計的資料庫稽查系統困難點的所在了。

另外，在一般的三層網路服務架構中，在使用者端與 UI 伺服器中間會存在一個防火牆，過濾一些可疑來源送過來的封包，如圖 5 所示。

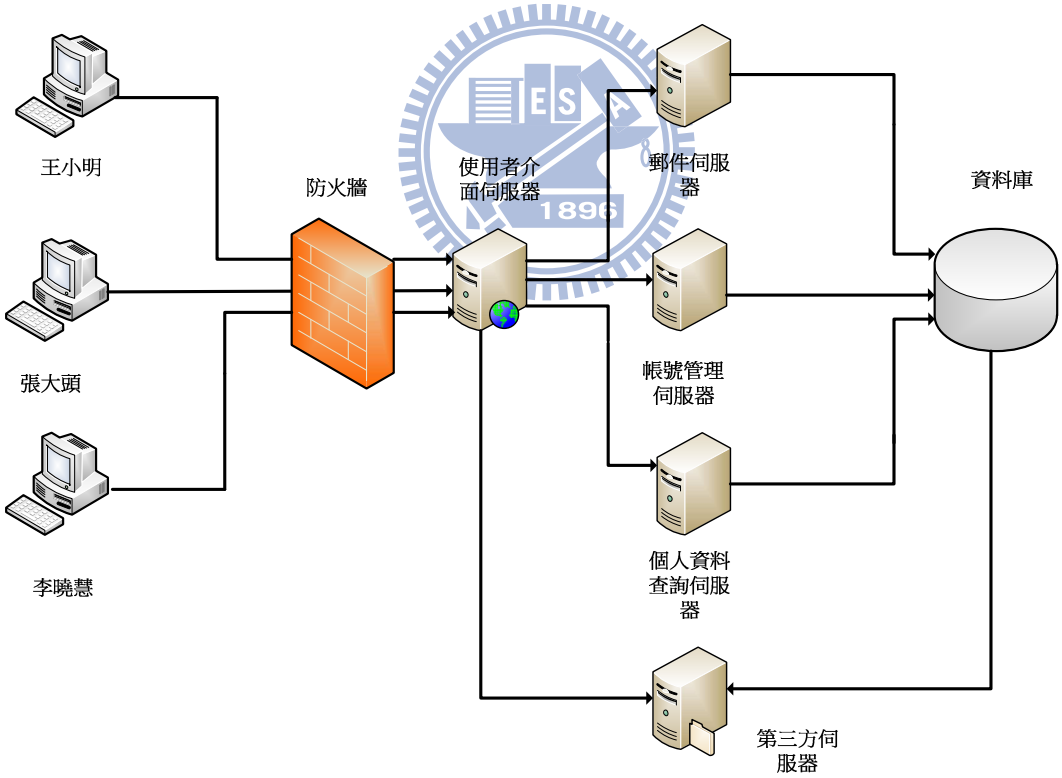
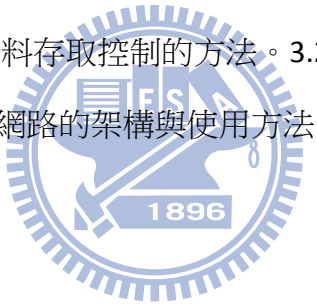


圖 5 在防火牆下的資料庫稽查系統

藉由這樣的方式，當使用者的服務要求送往 UI 伺服器時，很多沒有意義或是來自可疑來源的封包都已經被防火牆擋掉了，也因此當 UI 伺服器後續傳送 http 封包到第三方伺服器時，才不會多了很多沒有用的紀錄檔，而導致儲存空間的大量成長與後續 http 封包與 SQL 封包配對時造成大量的誤判情形。而由於資料庫接收的 SQL 封包都是即將執行的命令，因此在這部分，我們是採取只要資料庫一收到 SQL 指令，就全部送往第三方伺服器並加以記錄。藉由這樣的方式，我們在 http 封包與 SQL 封包配對時，就不會因為 UI 伺服器端傳送太多沒有意義的 http 封包給第三方伺服器，而導致後續的配對錯誤了。

第三章 相關研究

在一般資料庫的應用中，為了保持資料不受不相干人士的篡改，因此對不同的使用者常常給予不同的存取權限來達到資料的存取控制(Access Control)。除此之外，為了方便日後資料庫的稽查，因此也常在 AP 伺服器上紀錄每筆進來的 http 封包與相對應的 SQL 封包，然後在後續資料的分析時，利用一些機器學習或資料探勘的工具，從大量的資料中找出有用的資訊。常用到的工具有類神經網路、決策樹、貝氏分析器、支援向量機、基因演算法等，而類神經網路及決策樹又分別為生物資訊分析者與商業風險分析者所愛用，而為了達到生物資訊與商業風險分析的準確性，因此對於分析工具的效能上會更加要求，因此我們就類神經網路與決策樹做個介紹。3.1 說明資料存取控制的方法。3.2 介紹 AP 伺服器於資料庫稽查上的應用。3.3 說明類神經網路的架構與使用方法。3.4 說明決策樹的架構與使用方法。



3.1 資料存取控制的方法

在對於不同使用者採取資料存取控制時，一般我們採用自主存取控制(Discretionary Access Control; DAC)、強制存取控制(Mandatory Access Control; MAC)、角色基礎存取控制(Role Based Access Control; RBAC)等三種。

以自主存取控制來看，顧名思義就是使用者可以自行決定資料可以由哪些人來存取，像是 BBS 或是社交網路平台(如:臉書、噗浪等)的使用者可以決定跟自己相關的資訊(如:生日、手機號碼、相簿)可以給哪些朋友或是朋友的朋友存取。這樣的存取控制方式是非常自由的，但如果使用者沒有完善的設定，那麼使用者個人的隱私可能會在不自覺的情況下讓其他使用者一覽無遺。

而關於強制存取控制，主要可用 Bell-Lapadula 規則來探討。在 Bell-Lapadula 規則中，主要可分為主題(subject)與物件(object)，我們可以把使用者可以看成是主題，而把資料庫的資料看成是物件，除此之外，每個主題與物件都有不同機密等級的身分，依機密等級來看，可分為極機密(Top Secret)、機密(Confidential)、可信任(Confidential)、不重要(Unclassified)等四個安全級別。在這樣安全級別的分類下，主題與物件的存取控制遵守兩個原則：第一，不向上讀，也就是說機密等級較低的主題不能讀取機密等級較高的物件；第二，不向下寫，也就是說機密等級較高的主題寫資料時不能寫入機密等級較低的物件。第一個原則較好理解，因為在公司或是政府部門，如果機密等級較低的一般職員可以自由存取機密等級較高的公司機密的話，那麼資料外洩的情形將更為嚴重。而第二個原則可看成是職位較高的經理人不能把機密等級較高的公司決策放到一般員工都可以存取的資料庫裡面(機密等級較低)，不然這樣也會造成資料外洩事件。

而角色基礎存取控制，則是在設定那些使用者可以存取哪些資料時，利用擔任該職位(角色)的使用者是否牽涉到該業務(資料)來設定其是否有權限存取，因此有可能職位較高的使用者因為沒有牽涉到該業務的執行，而無法存取該資料的情形。(以上資料存取控制的內容修改及節錄自資料來源[12])

3.2 AP 伺服器上的資料稽查

一般在做小型的資料庫稽查時，都是直接在 AP 伺服器上紀錄每個網頁發出哪些資料庫指令，以便日後的追蹤比對。然而這樣的方式在政府部門或是公司的資料庫應用方面卻是行不通的，因為政府部門或公司為了日後較好維護系統，因此在 AP 伺服器與資料庫伺服器上往往是外包給不同專門的廠商，因此資料庫廠商基本上是無法得知政府部門或是公司架構的 AP 伺服器內容。就算資料庫廠商能事先知道該 AP 伺服器內容，那麼為了要能在不同 AP 伺服器上做紀錄，資料庫廠商必須針對每台 AP 伺服器上的每個網頁做分析及客製化，這樣不但曠日費時，而且也不符合經濟效益。因此，我們才會設計一套資料庫稽查系統，讓資料庫廠商在不知道AP伺服器架構的情況下，也能達到一定程度的資料庫稽查效果。(以上 AP 伺服器上資料稽查的內容修改及節錄自資料來源[15]、[16])

3.3 類神經網路

類神經網路顧名思義，其網路架構是模仿生物神經網路，整個網路可分為三個部分，分述如下：

(一)處理單元

或稱人工神經元，為類神經網路的基本組成單位。模型如圖 6 所示，而輸出值和輸入值間的關係可用下列函式表示：$Y_j = f(\sum Wij Xi - \theta j)$。

其中，i 為輸入層之神經元數，j 為輸出層之神經元數，Y_j 為模仿生物神經元模型的輸出訊號，f()為模仿生物神經元模型的轉換函數，是一個用以將從其他處理單元輸入的加權值轉換成處理單元輸出值的數學公式。Wij 是模仿生物神經元模型的神經元強度，又稱連結加權值，Wij 表示第 i 個處理單元對第 j 個處理單

元之影響強度。Xi 為模仿生物神經元模型的輸入訊號，θ_j： 模仿生物神經元模型的閥值。

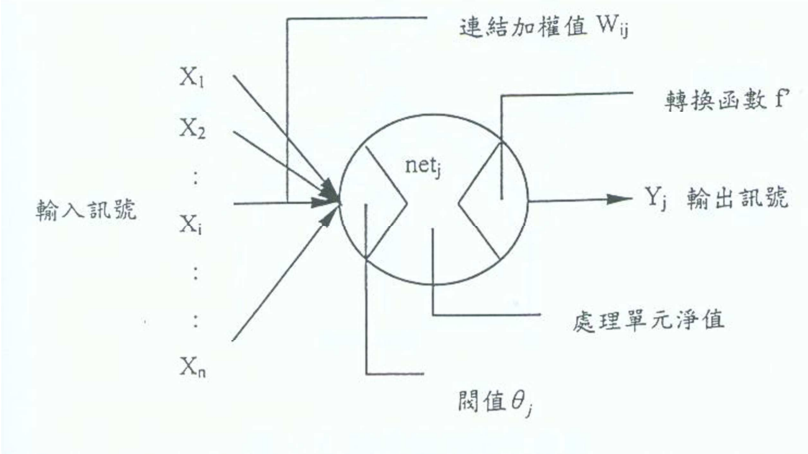


圖 6 處理單元示意圖

(二)層

若干個相同作用的處理單元之集合。依其作用可細分為正規化輸出、競爭化輸出以及競爭化學習。

(三)網路

幾個「層」進行堆疊集合，就成為了「網路」。如同在生物神經網路之中，神經元的強度可視為生物神經網路儲存資訊的所在，神經網路的學習即在調整神經結的強度。類神經網路各處理單元之間則以連接鍵互相連結，整個類神經網路的記憶就存放於這些連接鍵之中，以連接強度（權重）來表示。整個網路的基本架構如圖 7 所示。

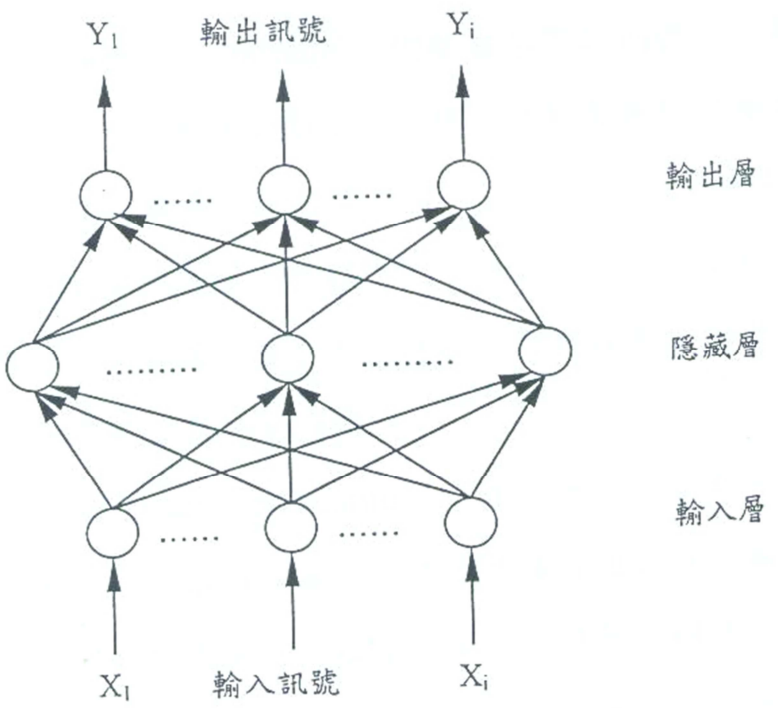


圖 7 類神經網路架構圖

類神經網路的運作可分為兩階段：(一)學習過程：網路依學習演算法，從樣本中學習已調整網路連結加權值，使網路的輸出盡可能和期望的輸出值一樣。若網路達到穩定的狀態時，則學習過程即可終止。(二)回憶過程：網路依回想演算法，以輸入資料決定輸出資料的過程。(以上類神經網路部分內容修改及節錄自資料來源[17]、[18])

3.4 決策樹

決策樹是一種預測模型，它可以找出一堆輸入值與後續輸出值的對應關係，而且樹中的節點代表某個對象，而每個分叉路徑則是代表該對象可能的屬性值。此外，每個葉節點則代表從對應根節點到該葉節點所經歷路徑表示對象的值。

決策樹學習也是資料探勘中一個普通的方法。在這裡，每個決策樹都呈現一種樹狀結構，它由它的分支來對該類型的對象依靠屬性進行分類。每個決策樹可以依靠對來源數據資料的分割進行數據測試，這個過程可以遞歸式的對樹進行修剪。當不能再進行分割或一個單獨的類可以被應用到某一分支時，遞歸過程就完成了，此外決策樹也可以依靠計算條件機率來構造。例如我們的來源資料(X, Y)為(x_1, x_2, x_3, …, x_k, y)中，相關變數 y 表示我們嘗試去找出的結果，而其他變數 x_1, x_2, x_3 等則是幫助我們達到目的的變量。

建立方法為

(1)以資料母群體為根節點。

(2)做單因子變異數分析等，找出變異量最大的變項作為分割準則。(決策樹每個葉節點即為一連串法則的分類結果。)

(3)若判斷結果的正確率或涵蓋率未滿足條件，則再依最大變異量條件長出分岔。

例如今天我們有個考慮天氣好壞與是否玩棒球的問題，我們事先詢問 23 個人在不同天氣狀況及相異濕度及是否颳風的情況下是否會玩棒球的情形，並加以分類。然後我們得知以下的結果：如果今天天氣是晴天或多雲，那麼人們常常會玩棒球，但只有少數人在雨天也會玩棒球。然後我們發現如果天氣晴天，但濕度高於 65%的話，人們也不會玩棒球，另外如果下雨天又有風的話那麼就一定不會有人玩棒球。那麼我們就可以建立出如圖 8 所示的決策樹了。(以上決策樹部分內容節錄、修改自資料來源[19]、[20])

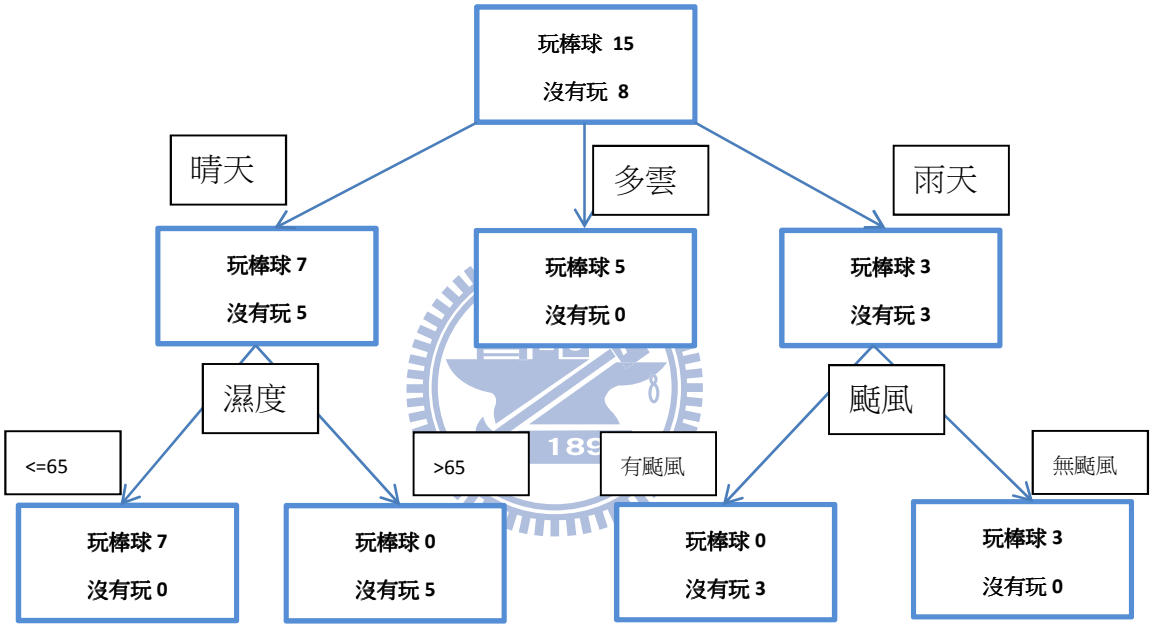


圖 8 決策樹的模型

第四章 系統設計

由於我們主要目的是要找出 UI 伺服器送出的 http 封包及 AP 伺服器送往資料庫端的 SQL 封包間的配對，找到最有可能配對成功的結果，進而找出該 SQL 封包是由哪個 AP 伺服器所送出的。因此在本章節，我們會討論到我們資料庫稽查系統的大體架構為何，及其細部的內容和實作方法，在 4.1 節我們介紹我們的大體架構，可分初步分類及進階分類兩個階段。在 4.2 節我們說明初步分類階段是如何分類的。在 4.3 節我們說明進階分類階段的實作方法及如何利用 Neural Network 來達成進階分類的功能。

4.1 資料庫稽查的架構

在我們設計的資料庫稽查系統中，主要可分為初步分類與進階分類兩個階段，初步分類階段主要是利用去除雜訊、對封包做排序等前置處理的方法，先取出可以用的資料，然後在利用一些背景知識及資料探勘的技術找出有用的資訊並加以統計，以供後續進階分類時的 Neural Network 使用。在處理完初步分類的階段後，接下來我們利用先前收集的統計資訊提供 Neural Network 使用，使進階分類能提供更精確的比對。而其中的初步分類階段及進階分類階段分別如圖 9 與圖 10 所示。

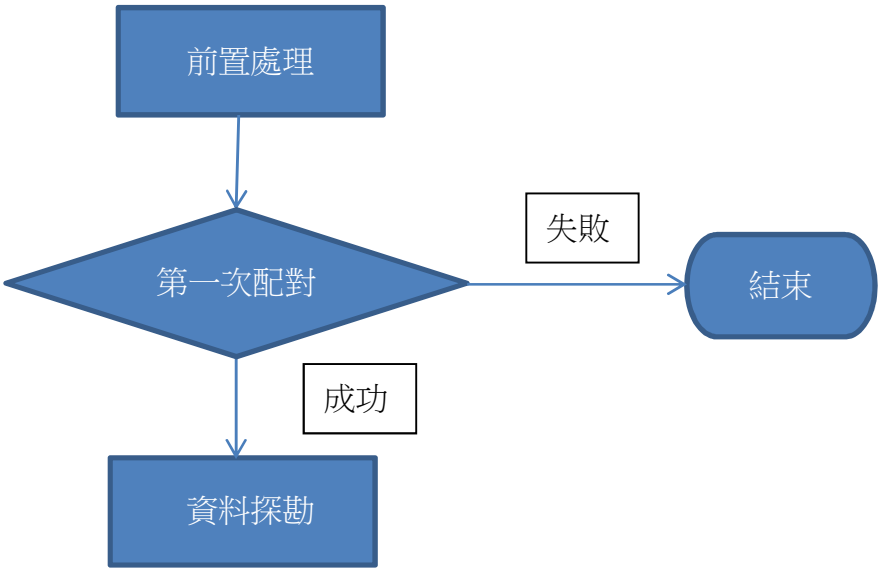


圖 9 初步分類階段的流程圖

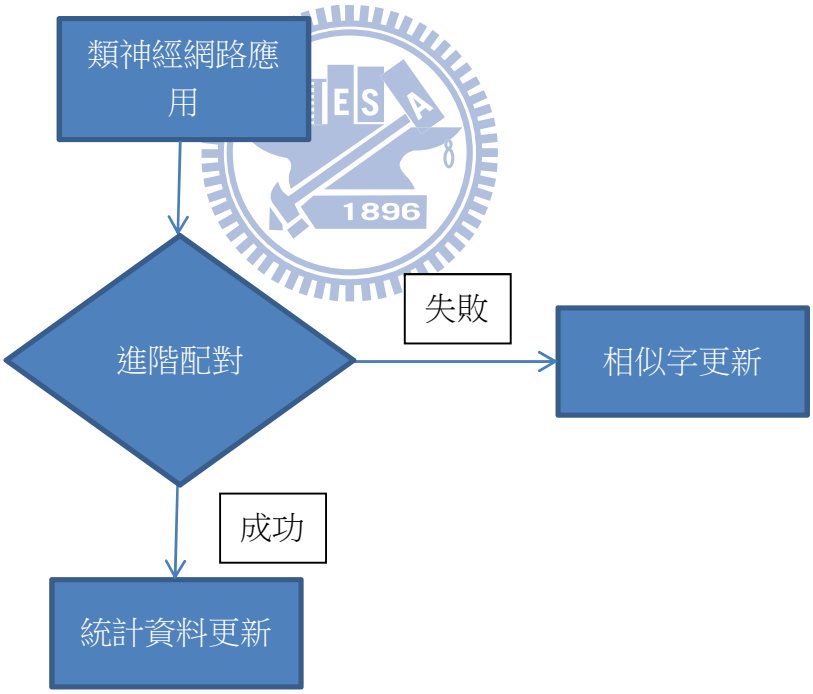


圖 10 進階分類階段的流程圖

4.2 資料庫稽查系統 – 初步分類階段

在此階段中主要是執行前置處理動作，然後進行第一階段比對，接著從比對完後的資訊找出統計資訊，並進行類神經網路的學習。

4.2.1 前置處理 – 處理封包雜訊

由於 http 封包或 SQL 封包在傳輸的過程中，可能因為一些原因，而導致封包在送達時的內容與一開始的內容有所差異，可能有多出一些數值或是缺少一些數值的情形，在此情況下，如果不先把雜訊處理好，我們後續的封包比對動作便會變得困難，因此我們一開始要做的事情就是處理封包雜訊問題，那我們如何處理呢？

以當傳送的是 http 封包為例，該封包內容為{a_1, a_2, a_3, a_4, a_5}，欄位 a_1、a_2、a_3、a_4 分別代表封包的時間郵戳、封包中的網頁 URI、封包的標籤名稱、封包的標籤數值，而欄位 a_5 代表封包中欄位 a_1 到 a_4 串接後的雜湊值。此外，a_1、a_2、a_3、a_4 為該封包最主要的部份。那麼封包送達後，只要 a_1、a_2、a_3、a_4 這四個欄位的值還在，那麼即使後面缺少了 a_5 的值，我們就把 a_5 視為空字串即可；若是 a_5 後有多出來的數值，那麼我們就把它們串接成一個數值，然後把該數值當成 a_5 欄位的值，如表 5 所示。

	時間郵戳	網頁 URI	標籤名稱	標籤數值	雜湊值	
原本封包						
	1279180201	ec-exit.asp	pp	6	TD698KwQA	
	1279180203	pos-pass-check.asp	B1	Login	TD6+S6wQA	
收到的 http 封包						
	1279180201	ec-exit.asp	pp	6	NULL	
	1279180203	pos-pass-check.asp	B1	Login	TD6+S6wQA	445
處理後的 http 封包						
	1279180203	pos-pass-check.asp	B1	Login		

表 5 出現雜訊的封包處理過程

4.2.2 前置處理 – 對封包進行排序

在處理完封包雜訊問題後，我們考量到封包在傳輸時，經由的傳輸路徑並不一定，因此常發生較早送出的封包卻比較晚送達的情形，因此在進行下一步前，我們先對所有的封包採取時間郵戳來做排序，在此我們利用 Quick Sort 來對封包做排序，排序前的 http 封包如圖 11 所示，而排序後的 http 封包如圖 12 所示。

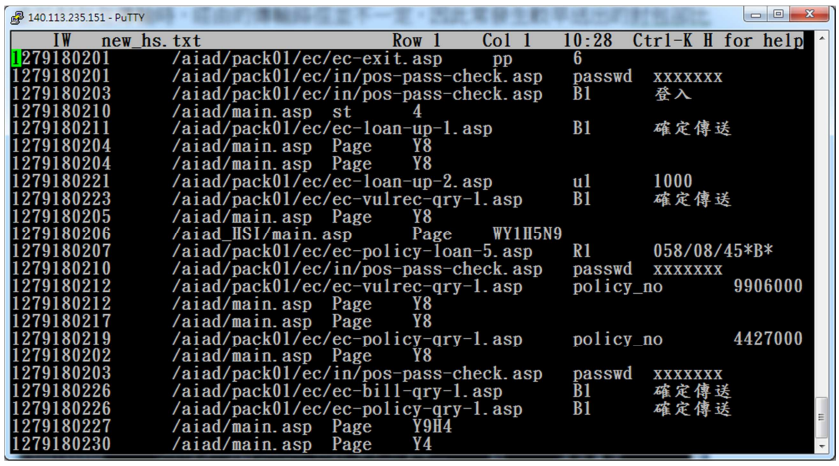


圖 11 依時間郵戳排序前的 http 封包

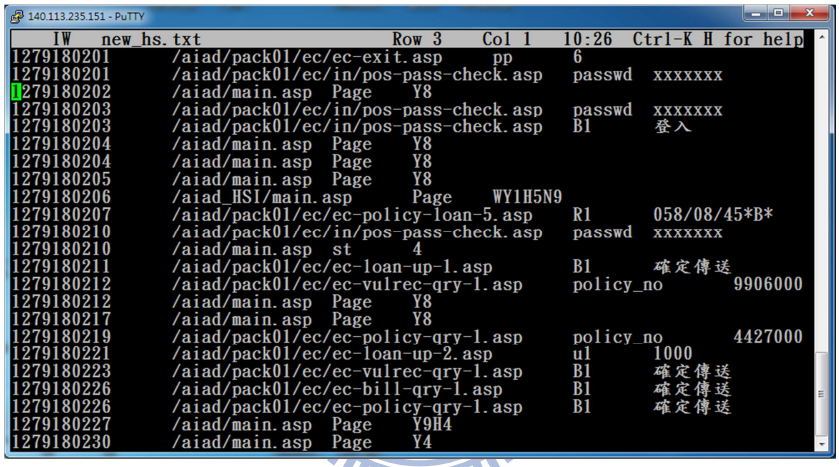


圖 12 依時間郵戳排序前的 http 封包

4.2.3 前置處理 – 建立相似字資料表

由於在做 http 封包與 SQL 封包的標籤名稱比對時，如果只單純用完整的標籤名稱比對，則很容易會遺失資訊。如表 6 與表 7 所示，若我們在標籤名稱的比對時只考慮到用 Policy_no 做比對，那麼在 SQL 封包中的標籤名稱 Policy 及 policy_no 將無法被認為是與 Policy_no 相同的比對，如此一來即使原本應該配對成功的兩筆 http 封包與 SQL 封包，也會因標籤取名的些微差異，而被誤判為不該是配對的 http 封包與 SQL 封包。

時間郵戳	網頁 URI	標籤名稱	標籤數值
1279180271	ec-policy-qry-1.asp	Policy_no	9442000
1279180401	ta-policy-qry2.asp	Policy_no	213

表 6 標籤名稱為 Policy_no 的 http 封包

時間郵戳	SQL 指令
1279180271	select insurance_type from polf where Policy = ?
1279180401	select a . exrt_source_ind from pldf a , polf b where a . plan_code = b . basic_plan_code and where policy_no = “213”

表 7 與標籤名稱為 Policy_no 的 http 封包配對的 SQL 封包

因此，我們在處理資料之前先將所有 http 及 SQL 封包中的變數名稱先取出，然後以人工的方式判斷哪些可能是相同的變數名稱，然後將之歸類，那麼在後續比對時，只要找到類似的變數名稱，便可將其認定為是同樣的意義，那麼就比較不會有誤判的問題了。如圖 13 所示

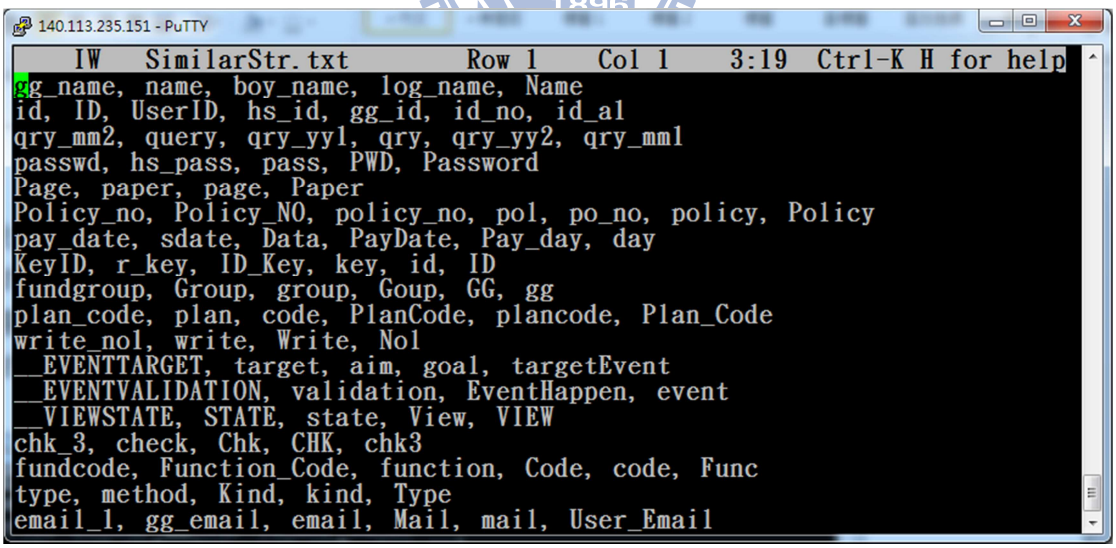


圖 13 標籤相似字的資料表

4.2.4 第一階段配對 – 時間郵戳的判斷

由於一般在使用瀏覽器瀏覽網頁時，若伺服器端太久沒回應的話，那麼瀏覽器會斷開該連線。加上 AP 伺服器只有在收到 UI 伺服器送來的 http 封包後， AP 伺服器才會傳送 SQL 封包給資料庫，所以 SQL 封包中的時間郵戳會比相對應的 http 封包時間郵戳來得晚。因此我們利用這兩個觀念來縮小找尋 SQL 封包可能配對的 http 封包範圍，用以達到節省比對時間及更能準確找到配對的目的。假設今天我們要比對的 SQL 封包其時間郵戳為 T_{SQL}，而可能比對到的 http 封包其時間郵戳為 T_{http}，那麼可能配對的 http 封包必須符合下列這兩個條件：

1. $T_{SQL} - T_{http} <= 10$
2. $T_{SQL} >= T_{http}$

以圖 14、圖 15 為例，圖 14 代表的是我們取出 SQL 封包的內容，圖 15 代表的是我們取出的 http 封包內容

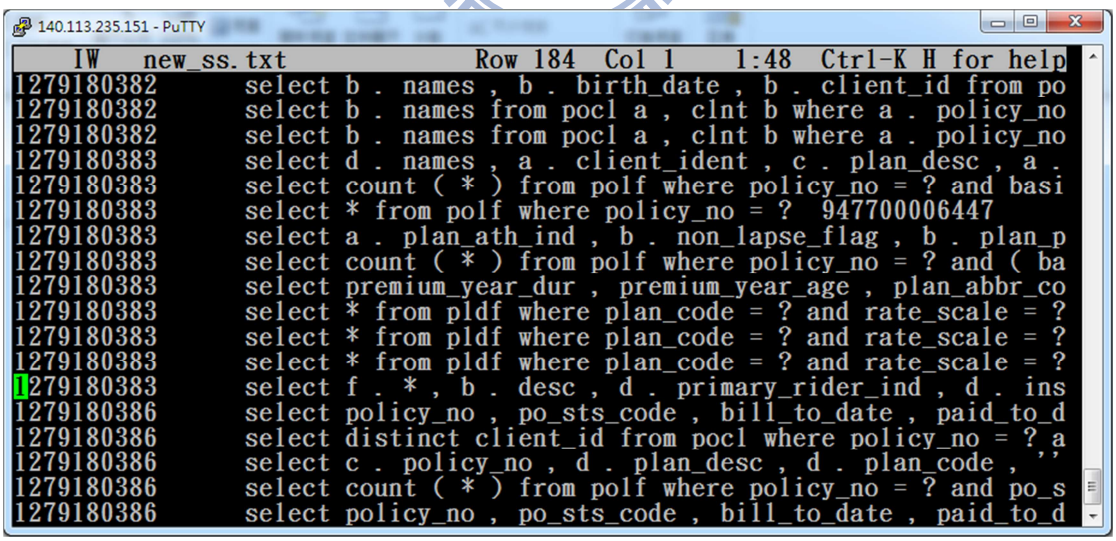


圖 14 我們要比對的 SQL 封包內容

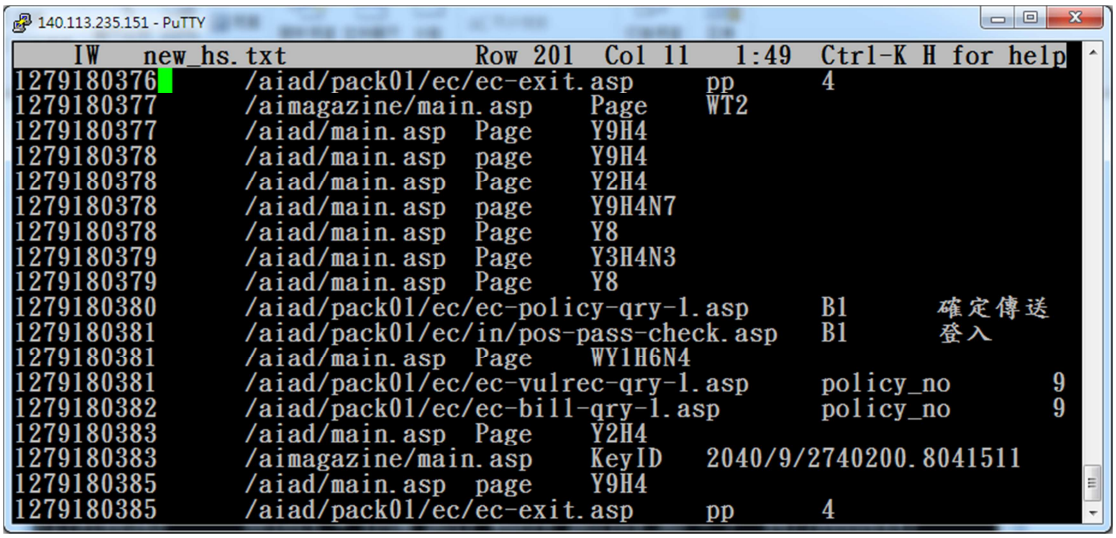


圖 15 可能比對成功的 http 封包內容

以上述 SQL 封包裡時間郵戳 1279180382 的 SQL 封包為例子，則可能比對符合的 http 封包其時間郵戳 T_{http} 必須符合 1279180382 - T_{http} <= 10 及 1279180382 >= T_{http} 這兩個條件，從上述的 http 封包範圍限縮後，我們發現 http 封包裡時間郵戳 1279180376 到 1279180382 的這些資料符合可能比對成功的 http 封包條件。

4.2.5 第一階段配對 – 關鍵字比對 & 計分系統

在前置處理完及縮小比對範圍後，接下來我們要進行第一次的 SQL 與 http 封包比對，在此我們利用兩個步驟來達成我們的目的。第一，利用關鍵字的比對，第二，利用計分系統來算出哪個 http 封包比較符合 SQL 封包的配對。

在關鍵字比對的部分，我們分成兩個步驟，第一個步驟是標籤名稱的相似字比對，第二個部分則是標籤數值的精確比對。在標籤名稱的比對部分，我們利用先前所建立的相似字資料表的部分做查詢，若 http 封包中的標籤名稱的值與 SQL 封包中擷取出的變數名稱是屬於相似字資料表中的同一類，那麼就表示它們是相同的，如表 8、表 9、表 10 所示。

Index	時間郵戳	網頁 URI	標籤名稱	標籤數值
1	1279180271	ec-policy-qry-1.asp	Policy_no	9442000
2	1279180401	ta-policy-qry2.asp	Policy_no	213

表 8 比對標籤相似字的 http 封包

索引值	時間郵戳	SQL 指令
1	1279180271	select insurance_type from polf where Policy = “9442000”
2	1279180401	select a . exrt_source_ind from pldf a , polf b where a . plan_code = b . basic_plan_code and where policy_no = “213”

表 9 比對標籤相似字的 SQL 封包

相似字資料表中查詢到 Policy_no 相似字的資料為

Policy_no	Policy_NO	policy_no	pol	policy	Policy

表 10 在相似字資料表裡查詢到的相似字資料

但在 http 封包標籤數值與 SQL 封包中變數數值的比對時，就必須是完整的比對，因為一般 http 封包中標籤數值若在 SQL 指令的變數數值中出現時，那麼兩者的值往往是一樣的。如表 8 與表 9 所示，http 封包中索引值為 1 的標籤數值與 SQL 封包中索引值為 1 的變數數值都是 9442000。

在比對完哪些 http 封包可能是 SQL 封包的來源後，接下來我們定義一個計分的公式，就是先為每一筆 http 封包都設定一個起始分數(預設是 0)，然後每一筆 http 封包找到與 SQL 封包相對應的標籤名稱時，我們就為該 http 封包加上 x 分，若找到與 SQL 封包相對應的標籤數值時，我們就為該 http 封包加上 y 分，最後取最高分者為我們認為最佳的配對，但若最高分者出現兩筆以上時，那麼我們就取 http 封包與 SQL 封包中時間郵戳最相近者為我們認為的最佳的配對。(預設 x 為 1，y 為 20)

例如我們要找表11所示的該筆內容，而經過時間篩選後的資料如表12所示，則經過分數計算後，我們得到如表 13 的結果。在表 13 中我們看到 http 第 3 筆與第 4 筆的資料都獲得 21 分，而且第 4 筆資料的時間郵戳距離我們要比對的 SQL 封包時間郵戳較為接近，因此我們選擇第 4 筆 http 封包為第一筆 SQL 找到的配對。

Index	時間郵戳	SQL 指令
1	1279180382	select insurance_type , po_sts_code from polf where policy_no = “990600009”

表 11 計分系統下的 SQL 封包

Index	時間郵戳	網頁 URI	標籤名稱	標籤數值
1	1279180376	/aimagazine/main.asp	Page	990600009
2	1279180377	pos-pass-check.asp	Policy	213
3	1279180377	ec-vulrec-qry-1.asp	Policy	990600009
4	1279180381	ec-policy-qry-1.asp	Policy_no	990600009

表 12 計分系統下的 http 封包

篩選後的 http 封包與時間郵戳 1279180382 的 SQL 封包經過標籤名稱與標籤數值比對後所到的分數表

http 的索引值	1	2	3	4
分數	20	1	21	21

表 13 計分系統下 http 封包與 SQL 封包配對的計分

在處理完第一次的 SQL 與 http 封包比對後，我們產生的資料欄位為 http 封包時間郵戳、SQL 封包時間郵戳、http 的網頁 URI 及 SQL 封包中的 SQL 指令，如圖 16 所示。

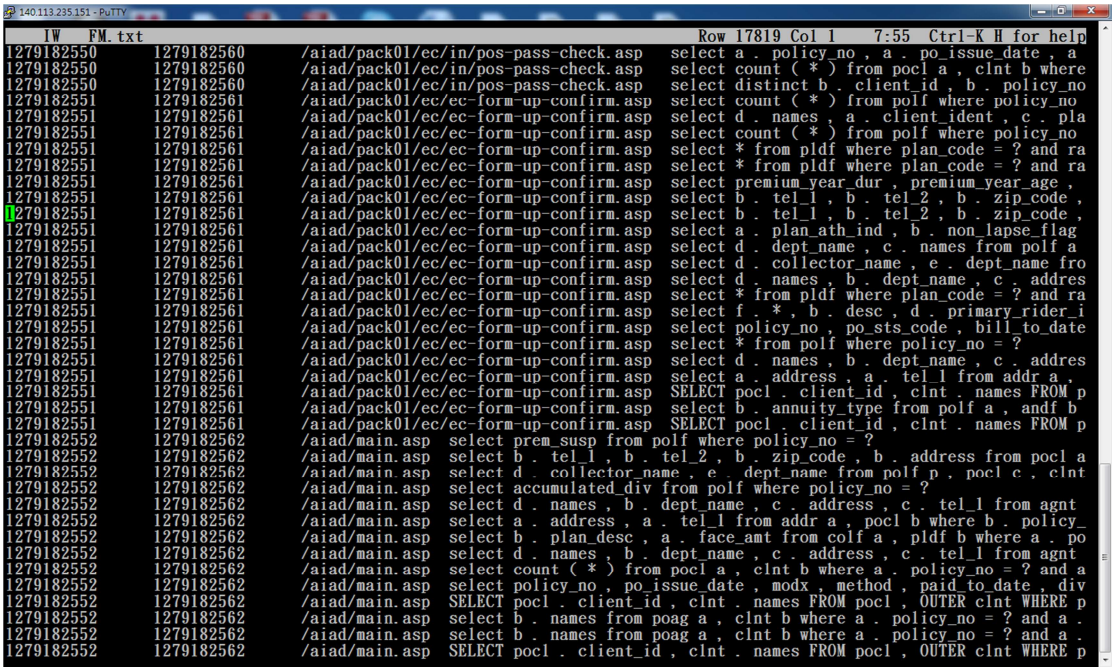


圖 16 處理完第一次比對後的結果

4.2.6 資料探勘

在第一次的比對時，有不少獲得最高分的 http 封包個數，事實上是超過一筆的，因此接下來我們要做資料探勘，從原本資料中找出較重要的資訊並加以統計整理，使這些超過一筆的最高分 http 封包在之後分析時能加以考慮這些資訊，以達到更精準的分析。

但我們要統計分析什麼樣的資訊呢？在此我們認為每個網頁中出現過哪些標籤名稱、每個標籤名稱的次數、每個標籤數值的次數、出現過哪些 SQL 操作指令、該 SQL 指令的出現次數、用過那些資料庫種類，及這些標籤名稱、標籤數值、SQL 操作指令、資料庫種類的出現頻率等等，是比較重要的資訊。

我們統計每個網頁含那些標籤及每個標籤出現過的次數結果如圖 17 所示，每個網頁含那些 SQL 操作指令及每個 SQL 操作指令在網頁中出現次數如圖 18 所示，

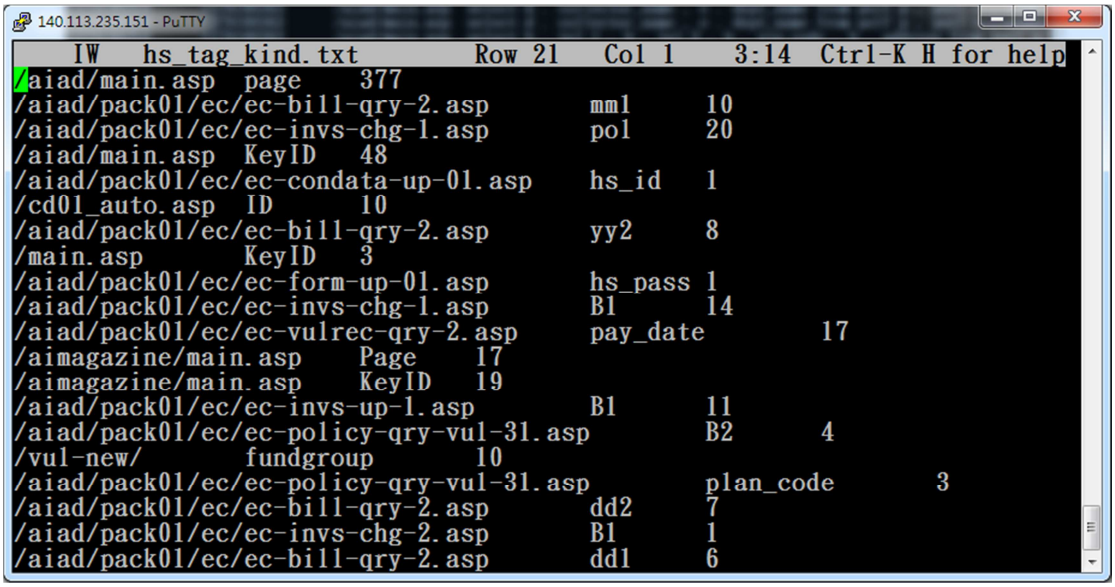


圖 17 每個網頁含有的標籤名稱及該標籤出現次數

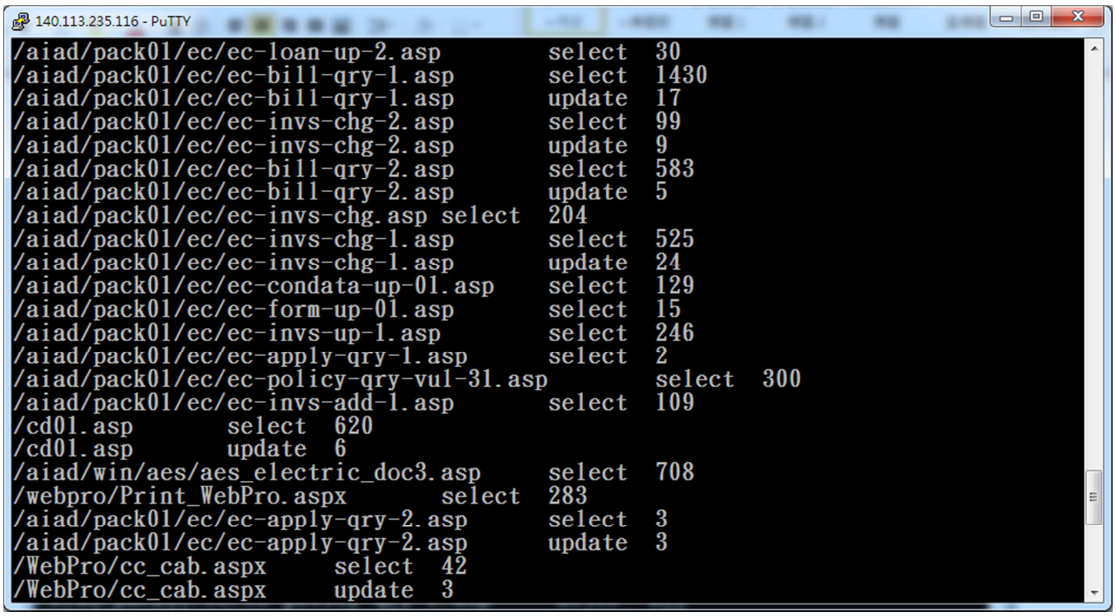


圖 18 每個網頁含有的 SQL 操作指令及該指令出現次數

4.3 進階分類的階段

在初步分類結束之後，就可以進行我們的進階分類了，在這個階段一開始先對初步分類階段所找出的統計資訊利用類神經網路加以訓練，然後針對初步分類階段中第一次比對時，所遇到獲得最高分配對筆數超過一筆的情形，接著進行 Neural Network 的測試然後獲得新分數，再比較的這些配對獲得的新分數高低，以最高分者為我們認為最佳的比對。

4.3.1 Neural Network 訓練過程

在做完資料探勘的動作後，我們會得到一些重要的統計資訊，接下來我們用 Neural Network 來對我們的資料做訓練的動作。我們採用的是倒傳遞的神經網路，然後輸入的屬性為 SQL 封包與 http 封包時間郵戳差值，及另外的 8 個屬性，分別為該 http 封包網頁 URI 欄位的網頁中，出現過標籤名稱的次數、標籤數值的次數、SQL 操作指令及用過的資料庫種類等出現次數的頻率，及這些標籤名稱、標籤數值、SQL 操作指令、資料庫種類的頻率。

在得到這些結果之後，我們再進行把輸入資料正規化的動作，在時間郵戳差值的部分，我們利用表 14 中的轉換公式來把時間郵戳差值變成 0 到 1 之間的數值。而剩下的 8 個屬性，則是直接算出其相對比例。

時間郵戳差值	正規化的結果
X	(10 - x) / 10

表 14 時間郵戳差值的轉換公式

而輸出值的部分，我們則是以第一階段配對成功的 SQL 封包與 http 封包中是否含有相同的標籤名稱與標籤數值來做取值，如表 15 所示。

無相同標籤名稱、數值	有相同標籤名稱或數值	有相同標籤名稱與數值
0.0	0.5	1.0

表 15 輸出值的計算方式

在經過第一階段比對後，我們得到表 16 所示的一筆 http 封包與 SQL 封包配對。

SQL 封包	select d.names , b.dept_name , c.address , c.tel_1 from …
對應的 http 封包網頁 URI	/aiad/pack01/ec/ec-policy-qry-1.asp
時間郵戳差值	1279180587 – 1279180587 = 0
相似標籤名稱	policy_no
相同標籤數值	無
相同的資料庫指令	select
相同的資料庫	life

表 16 第一階段比對成功的一筆資料

接著我們查詢其相關的統計資料，以便進行正規化的動作。而圖 19 為紀錄網頁名稱、該網頁出現的標籤名稱、該標籤出現次數、該網頁的標籤種類數、該網頁所有標籤出現次數總和的結果。圖 20 為記錄網頁名稱、該網頁出現的資料庫指令、該指令出現次數、該網頁的資料庫指令種類數、該網頁所有資料庫指令出現次數總和的結果。

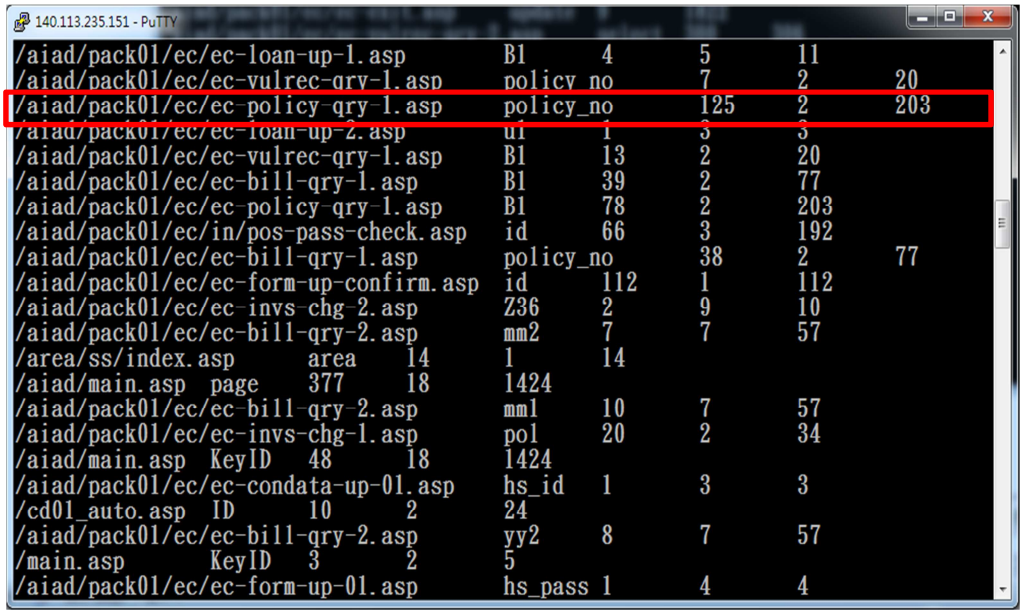


圖 19 每個網頁含有標籤名稱的統計資料

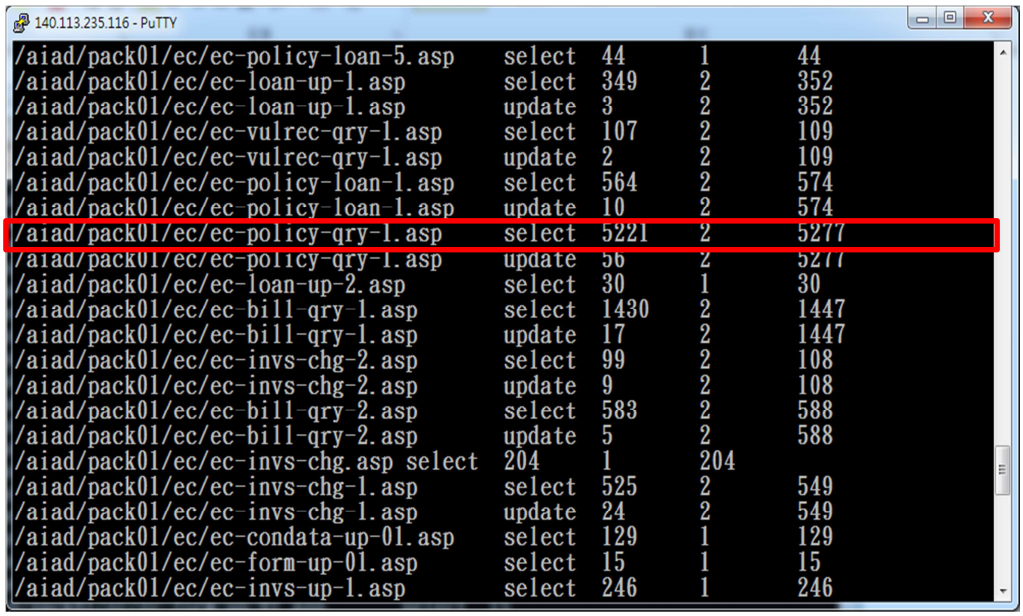


圖 20 每個網頁含有 SQL 操作指令的統計資料

在經由上述查表動作，我們得到的類神經網路訓練資料如表 17 所示，

輸出值	時間差	標籤名稱次數	標籤數值	資料庫指令	資料庫	標籤名稱種類	標籤數值種類	資料庫指令種類	資料庫種類
0.5	(10-0)/10	125/203	0.0	5221/5277	1/1	1/2	0.0	1/2	1/1

表 17 類神經網路訓練的資料格式

另外，在我們使用的類神經網路部分，預設有一個隱藏層，隱藏層內有 200 個隱藏神經元，還有一個輸出神經元，訓練的次數為 2000 次，以及訓練的樣本數有 10000 個。其訓練次數與訓練出來的誤差結果如圖 21 所示，圖 21 前面的數字表示回合數(每 10 回合為一單位)，後面數字表示誤差值，我們可以看出當訓練到越後面，其誤差的結果改變地幅度卻不大，因此有達到收斂的目的。而圖 22 為我們訓練出來的輸入層與隱藏層間的權重係數表。

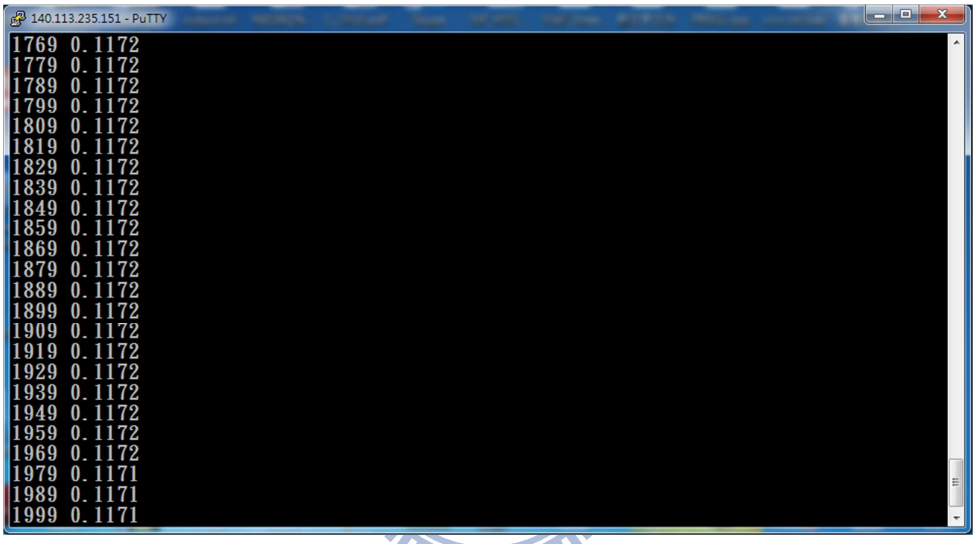


圖 21 Neural Network 訓練後的每 10 回合數及其相對應誤差結果

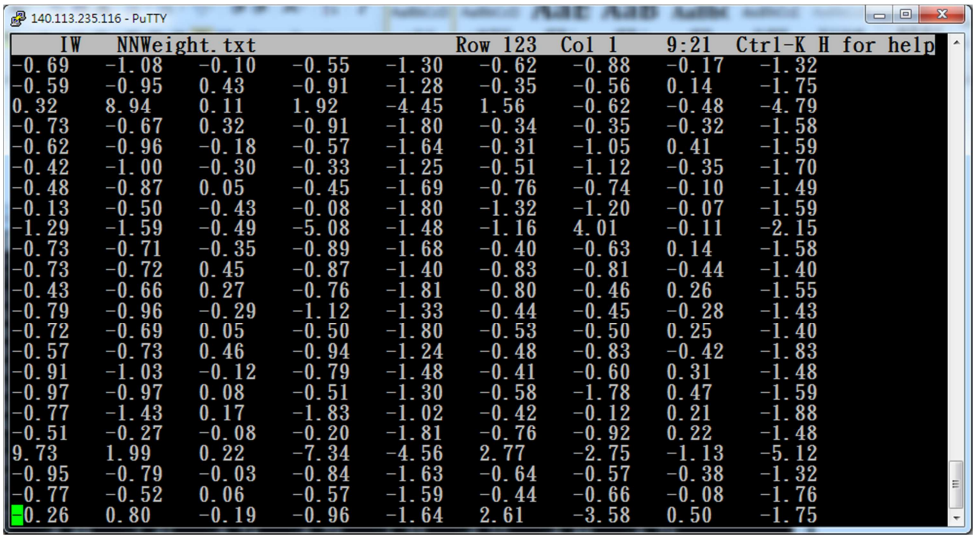


圖 22 類神經網路訓練後輸入層與隱藏層間的權重表

4.3.2 進階比對過程

我們要比對的 SQL 封包如表 18 所示，而表 19 則是在學習階段中第一次比對後獲得最高分的 http 封包在進階分類階段中經過 Neural Network 測試後的得分結果。

時間郵戳	SQL Statement
1279180208	select userInfo from User where Password = “1234567”

表 18 進階分類階段中比對的 SQL 封包

時間郵戳	AP 伺服器中的網頁 URI	AP 伺服器中的網頁標籤名稱	AP 伺服器中網頁標籤數值	第一次配對分數	進階配對分數
1279180206	ec-login.asp	passwd	1234567	21	0.82
1279180206	inpos-pass-check.asp	passwd	1234567	21	0.61

表 19 回憶階段中經過 Neural Network 測試後的 http 封包

由表 19，我們得知網頁 URI 為 inpos-pass-check.asp 的 http 封包比較符合我們表 18 中 SQL 指令的配對。

另外在配對完之後，我們會從配對成功的 http 封包及 SQL 封包中找出每個網頁中所含的資訊，然後進行更新，使系統能藉由更多的資訊統計而達到更精準的目的。而我們統計的資訊包含該 http 封包網頁 URI 中的網頁中出現過哪些標籤名稱、每個標籤名稱的次數、每個標籤數值的次數、出現過哪些 SQL 操作指令、該 SQL 指令的出現次數、用過那些資料庫種類，及這些標籤名稱、標籤數值、SQL 操作指令、資料庫種類的出現頻率等等。

第五章 實驗過程與結果討論

在這一章，我們要討論我們的實驗結果。5.1 說明我們的實驗的硬體規格及使用軟體，此外我們會討論實驗中耗用的 CPU 資源等。5.2 討論我們對於實驗中所耗費的時間及配對準確性的結果。

5.1 實驗環境及實驗時耗用資源

在實驗的部分，我們使用 Intel 雙核心 2.5GHz CPU(E5200)、兩條金士頓 2Gb 記憶體(DDR 2)、作業系統為 64 位元的 Microsoft Windows 7 平台，然後採用 Microsoft Visual Studio 2010 為我們的測試環境。我們分別對我們實驗中的初步分類階段中的第一次配對部分、類神經網路訓練部分及整個流程(扣除類神經網路訓練部分)的運作，計算其 CPU 使用率，結果如表 20 所示。

測試部分	第一次配對	類神經網路訓練	整個流程(扣除訓練部分)
CPU 使用率	28% - 33%	49%-54%	29% - 35%

表 20 不同測試部分的 CPU 使用率

由以上結果，我們得知 CPU 使用量最大的時候是在類神經網路訓練的時候，除此之外的其他步驟 CPU 使用率會比訓練階段少 20%以上的 CPU 使用率，而訓練的步驟通常只在我們一開始的階段，因此這樣的結果是我們可以接受的。

5.2 實驗耗費的時間及配對的準確度

除了上述 CPU 使用率的測試外，我們還分別對利用初步分類階段中第一次配對出來的結果與花費時間與利用跑完整個流程所得出的結果與花費時間做個分析。(在計算整個流程所花費的時間不包含 Neural Network 訓練時所花的時間)

利用初步分類階段中第一次配對所花費時間如圖 23 所示，利用跑完整個流程所花費時間(不計訓練時間)如圖 24 所示。

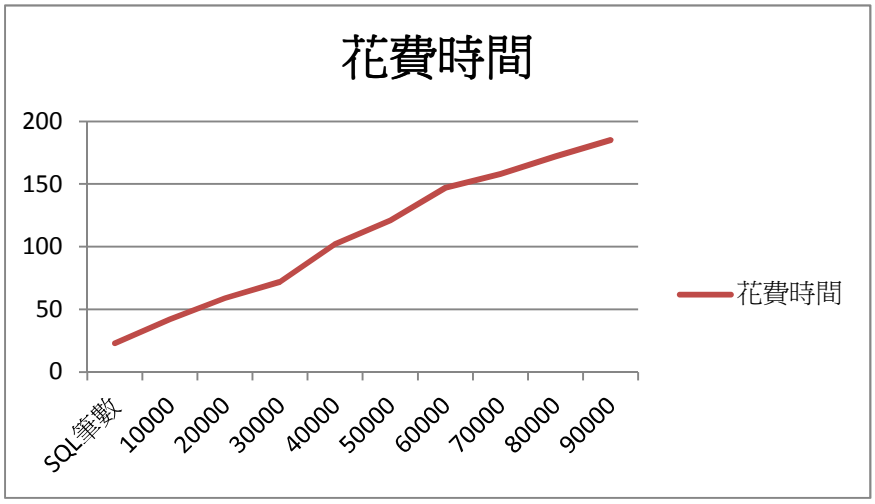


圖 23 第一次配對所花費時間

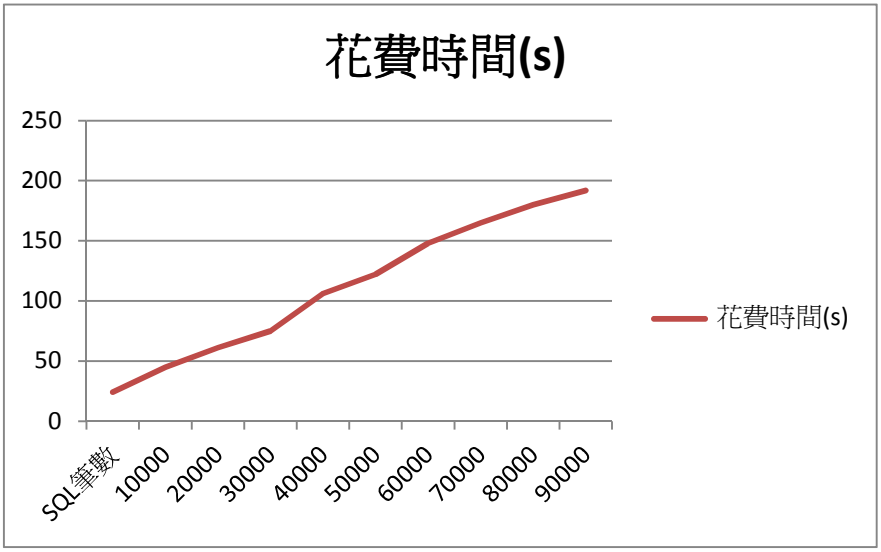


圖 24 跑整個流程所花費時間(不計算訓練時間)

我們發現採用 Neural Network 的方法來做 SQL 封包與 http 封包配對所耗費的時間，比只用初步分類階段中第一次配對的方法來做比對所花時間沒有多很多，這是因為 Neural Network 在學習的過程中會花費很多時間，但是使用在後續的比對上，所花的時間是非常少的，因此才會產生這種現象。

而圖 25、圖 26 分別表示在利用初步分類階段中第一次配對與跑完整個流程的情況下所配對出 http 封包與 SQL 封包的準確性。

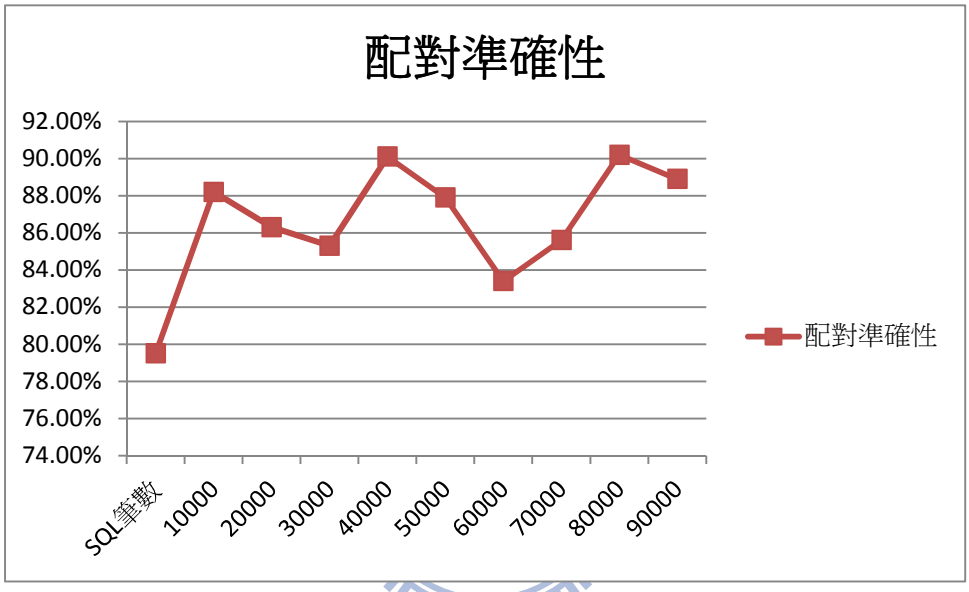


圖 25 用第一次配對方法所達到的準確性

在只採用第一次配對方法，而沒有使用 Neural Network 來做分類的情況下，我們發現隨著資料量的成長，比對的精準度卻沒有隨之成長。這是因為隨著資料量的成長，但我們卻沒有進行相關資料的收集與統計，因此在不知道每個網頁含有哪些資訊量的情況下，包含他們擁有那些標籤名稱、在該網頁中出現哪些標籤數值、哪些資料庫指令及哪些與之相關的資料庫變數，除此之外還包含上述這些屬性的出現次數及出現頻率等，因此準確性才沒有隨著資料量成長而跟著增加。

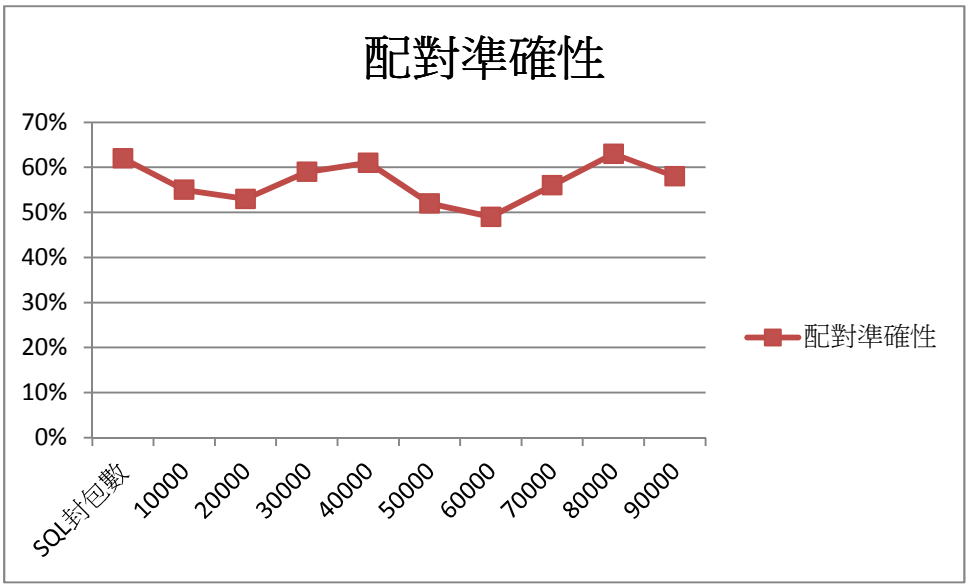


圖 26 在跑完整個流程下的配對準確性

在利用 Neural Network 的情況下，我們發現比對出來的結果並沒有比單純使用第一次配對方法的比對結果來得好的，甚至有下降的趨勢，這大概是因為我們在做 Neural Network 訓練的時候，參數沒有調整好，才使得雖然最後結果有收斂，但卻沒有收斂到一個較好的結果，也因此才會導致雖然用了類神經網路但最後配對結果卻沒有我們預期中的好。

第六章 結論

6.1 討論與結論

在本論文中，我們討論到如何利用 Neural Network、資料探勘及相關背景知識來找出每筆 SQL 封包相對應的 http 封包，進而找出每筆 SQL 封包的來源是那些 AP 伺服器。我們在儘量不變更三層式網路服務伺服器架構，及不考量 AP 伺服器的網頁是由何種網頁程式寫成的情況下，設計了一套資料庫稽查的方法。之後，我們在找 http 封包與 SQL 封包間的配對時，用了一些背景知識的處理方式，來獲得方便後續配對的 http 及 SQL 封包資訊。在獲得上述的 http 與 SQL 封包資訊後，我們再利用類神經網路來協助進階找出 http 封包與 SQL 封包間的配對。

雖然利用本文中提到的方法可以達到資料庫稽查的效果，但這樣的方法在現今資訊量龐大、網路服務掛帥的情況下，仍需要其他技術的支援使演算法效能部分能更加提升。像是相似字比對的部分在未來累計的種類與數目變多後，或許我們可以從電腦病毒比對或生物資料庫比對的技術中找尋更好的比對演算法，使得系統在面對更多的服務時也能達到不錯的效果。

然而我們在實驗中所提出的找尋 SQL 封包來源的方法，有部份都是參考一般伺服器在傳送封包時所產生的現象，進而擬定可以處理的方法，未來如果有特別針對像是沙賓法案或是個資法的部分，或許我們可以再依照這些需求提出額外的處理步驟，使得找尋 SQL 封包來源及後續資料庫稽查的精準度上能再提升。

6.2 未來工作

在資料庫的稽查的這個任務中，除了要先把發送每筆 SQL 封包的 AP 伺服器找到外，接下來就是要辨別在這些SQL封包中，那些可能是惡意或是違法的操作，例如公司內部非會計也不是人資部門的人士卻大量搜查其他人的資料，或是非上班時間卻使用資料庫從事公務行為等等，這些動作雖然不一定就是資料外洩的非法操作行為，但資料外洩卻往往藏在這些資料庫操作中。然而諸如此類可被歸納在可疑資料庫操作的行為卻不勝枚舉。因此如何定義這類可能造成資料外洩的行為，及是否能利用資料探勘和類神經網路的技術對資料安全領域進行更上一層的提升，便成為我們接下來最重要的議題了。

參考文獻

[1] Kang Li, Zhenyu Zhong, and Lakshmish Ramaswamy, “Privacy-Aware Collaborative Spam Filtering,” IEEE Transactions on Parallel and Distributed systems, Vol. 20, No. 5, Pages 725 – 739, May 2009.

[2] Zhenyu Zhong and Kang Li, “Speed Up Statistical Spam Filter by Approximation,“ IEEE Transactions on Computers, Vol. 60, No. 1, Pages 120 – 134, January 2011.

[3] Z. Bar-Yossef and S. Rajagopalan, “Template Detection via Data Mining and Its Applications,” In Proceedings of the 11th International Conference on World Wide Web, Pages 725 – 739, May 2002.

[4] M. Fisher, S. G. Elbaum, and G. Rothermel, “Dynamic characterization of web application interfaces,” In Proceedings of the 10th Fundamental Approaches to Software Engineering, Pages 260 - 275, May 2007.

[5] Y. Xie and A. Aiken, “Static detection of security vulnerabilities in scripting languages,” In Proceedings of the 15th conference on USENIX Security Symposium, Vol. 15, No.13, Pages 179 – 192, July – August 2006.

[6] Z. Zhong, L. Ramaswamy, and K. Li, “Alpacas: A Large-Scale Privacy-Aware Collaborative Anti-Spam System,” In Proceedings of the the 27th IEEE International Conference on Computer Communications, Pages 556 – 564, April 2008.

[7] T. Meyer and B. Whateley, “SpamBayes: Effective Open-Source, Bayesian Based, Email Classifications,” In Proceedings of the 1st Email and Anti-Spam Conference, July 2004.

[8] William G. J. Halfond, Alessandro Orso, and Panagiotis Manolios, “WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware Evaluation,” IEEE Transactions on Software Engineering, Vol. 34, No. 1, Pages 65 – 81, January 2008.

[9] Peter Likarish, Eunjin (EJ) Jung and Insoon Jo, “Obfuscated Malicious Javascript Detection using Classification Techniques,” In Proceedings of the the 4th International Conference on Malicious and Unwanted Software, Pages 47 – 54, October 2009.

[10] Vassilios S. Verykios, Ahmed K. Elmagarmid, Elisa Bertino, Yucel Saygin, and Elena Dasseni, “Association Rule Hiding,” IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 4, Pages 434 – 447, April 2004.

[11] Chi-Yao Tseng, Pin-Chieh Sung, and Ming-Syan Chen, “Cosdes: A Collaborative Spam Detection System with a Novel E-Mail Abstraction Scheme,” IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 5, Pages 669 – 682, May 2011.

[12] Elisa Bertino, and Ravi Sandhu, “Database Security—Concepts, Approaches, and Challenges,” IEEE Transactions on Dependable and Secure Computing, Vol. 2, No. 1, Pages 2 – 19, January-March 2005.

[13] Kieyzun, A., Guo, P.J., Jayaraman, K., Ernst, M.D., “Automatic Creation of SQL Injection and Cross-Site Scripting Attacks,” In Proceedings of the 31st IEEE International Conference on Software Engineering, Pages 199 – 209, May 2009.

[14] Shay Artzi, Adam Kiezun, Julian Dolby, Frank Tip, Danny Dig, Amit Paradkar, and Michael D., “Finding Bugs In Dynamic Web Applications,” In Proceedings of the 2008 international symposium on Software testing and analysis, Pages 261 – 272, July 2008.

[15] Tao Li and Chao Liu, “Data ‘Audit’ Research Based on The Accounting Information System,” In Proceedings of the 2010 International Conference on E-Business and E-Government, Pages 2416 – 2419, May 2010.

[16] Wei Chen, Wally J. Smieliauskas, and Si-fengLiu, “Performance Assessment of Online Auditing in China from the Perspective of Audit Cost Control, “ In Proceedings of the 2010 IEEE International Conference on Systems Man and Cybernetics, Pages 833-837, October 2010.

[17] 類神經網路模式應用與實作，葉怡成編著，(初版)2003 年，儒林出版社。

[18] 類神經網路與模糊控制理論-入門與應用，王進德編著，(初版)2006 年，全華圖書公司。

[19] 資料探勘原理與技術，張云濤、龔玲編著，(初版)2007 年，五南圖書公司。

[20] 資料採掘理論與實務規劃手冊，孫惠民編著，(初版)2007年，松崗出版社。