行政院國家科學委員會專題研究計畫　成果報告

子計畫二：NBEN 北區網路安全建置與處理

計畫類別：整合型計畫
計畫編號：NSC92-2219-E-002-018-
執行期間：92 年 05 月 01 日至 93 年 04 月 30 日
執行單位：國立臺灣大學資訊工程學系暨研究所

計畫主持人：賴飛羆
共同主持人：張瑞雄
計畫參與人員：林振群、楊子翔、黃俊穎、蔡林峻、洪啟仁、戴上為、陳澤世、吳胤鋒、林曉偉

報告類型：完整報告

處理方式：本計畫可公開查詢

中　華　民　國 93 年 6 月 7 日

行政院國家科學委員會補助專題研究計畫 ☑ 成 果 報 告
☐期中進度報告

國家寬頻實驗網路(NBEN)網路安全建置與實驗計畫

子計畫二：NBEN 北區網路安全建置與處理

計畫類別：☐ 個別型計畫　☑ 整合型計畫
計畫編號：NSC 92－2219－E－002－018－
執行期間：　92　年 5 月 1 日至　93　年 4 月 30 日

計畫主持人：　賴　飛　羆　　教授
共同主持人：　張　瑞　雄　　教授
計畫參與人員：　林振群、楊子翔、黃俊穎、蔡林峻、洪啟仁、戴上為
陳澤世、吳胤鋒、林曉偉

成果報告類型(依經費核定清單規定繳交)：☐精簡報告　☑完整報告

本成果報告包括以下應繳交之附件：
☐赴國外出差或研習心得報告一份（附件一）
☐赴大陸地區出差或研習心得報告一份
☐出席國際學術會議心得報告及發表之論文各一份
☐國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、
列管計畫及下列情形者外，得立即公開查詢
☐涉及專利或其他智慧財產權，☐一年☐二年後可公開查詢

執行單位：國立台灣大學資訊工程學系暨研究所
國立東華大學資訊工程學系暨研究所
中　華　民　國　93　年　6　月　7　日

摘要

隨著網路的普及化及頻寬的增加，網際網路(Internet)已成爲各種電子裝置間最重要的通訊管道。跟著而來的是，有心人可以利用系統的漏洞，設計不良的通訊協定，以及人爲的疏失，侵入某些連結網路的特定目標或是加以癱瘓，不論其目的爲何，這類攻擊如果成功，往往造成大量的資料損失，或是使系統正常工作停擺。本子計畫針對務阻斷攻擊(DoS attacks, Denial of Service attacks)和分散式服務阻斷攻擊(DDoS attacks, Distributed Denial of Service attacks)的攻擊來源追蹤及攻擊封包分流偵測技術進行研究，我們成功地發展了記錄式(logging-based)封包追蹤系統及訊息式(messaging-based)封包追蹤系統，以及靜態及 round-robin 式流量分流(Traffic splitting)系統。

關鍵詞：分散式服務阻斷攻擊、記錄式封包追蹤系統、訊息式封包追蹤系統、靜態流量分流系統、round-robin 式流量分流系統。

Abstract

With the popularity and availability of the Internet, it has become one of the most important communication channels between various electronic devices. However, malicious users can make use of various system loopholes, ill-devised protocols, and negligence caused by managements to hack into or paralyzed target systems via connected networks. These attacks usually result in the disclosure of privacy and confidential data, or inability to sustain normal services. In our project, we focused on DoS(Denial of Service) attacks and DDoS(Distributed Denial of Service) attacks. Our research topics included how to trace the sources IP addresses of such attack packets and how to apply traffic splitting techniques to improve the detection of such attackes. We developed a logging-based IP-traceback system and a messaging-based IP-traceback system. We also deployed a traffic splitting system with static traffic splitting rules and round-robin traffic splitting rules to enable the detection of more attack traffic.

Keywords: DDoS attacks, logging-based IP-traceback, messaging-based IP-traceback, static traffic splitting, round-robin traffic splitting.

目錄

一、前言

隨著網路的普及化及頻寬的增加，網際網路(Internet)已成爲各種電子裝置間最重要的通訊管道。跟著而來的是，有心人可以利用系統的漏洞，設計不良的通訊協定，以及人爲的疏失，侵入某些連結網路的特定目標或是加以癱瘓，不論其目的爲何，這類攻擊如果成功，往往造成大量的資料損失，或是使系統正常工作停擺。

在網際網路上常出現的攻擊，大約可以分爲兩種:

- 入侵式攻擊(Intrusion attacks):

成功的入侵目標系統後，可以使用該系統的資源，甚至是控制該系統。一般而言，入侵者通常會竊取被入侵系統的資料庫。入侵主要是攻擊系統程式，如 WWW 伺服器，資料庫伺服器等的漏洞，或是利用人爲操作的疏失，例如不小心執行帶有病毒的程式，而將攻擊者的程式(病毒，特洛伊木馬等)啓動運作。

- 頻寬式攻擊(Bandwidth attacks):

而頻寬式攻擊則是，攻擊者無需入侵其攻擊目標，但仍然可以使這整個系統無法或幾乎無法提供正常的服務。其攻擊的方式，則是破壞連結的網路(例如攻擊路由器)，針對 TCP/IP 協定的弱點，以大量的封包來加重攻擊目標的負荷，或使其停止，或是機乎無法對一般使用者進行服務。常見的服務阻斷攻擊(DoS attacks, Denial of Service attacks)和分散式服務阻斷攻擊(DDoS attacks, Distributed Denial of Service attacks)都屬於此類。

子計劃二由台灣大學與東華大學共同合作進行，台大負責的部分是針對服務阻斷攻擊(DoS attacks, Denial of Service attacks)以及分散式服務阻斷攻擊(DDoS attacks, Distributed Denial of Service attacks)進行研究，主要的重心放在如何追蹤攻封包的來源。當DoS/DDoS攻擊發生後，如能正確地追蹤攻擊來源，便可以有效地對此類攻擊進行處理，常用的方法包括阻斷(blocking)，限流(rate limiting)，入口過濾(ingress filtering)等，這些方法通常可以有效地將DoS/DDoS攻擊損害控制在可以接受的範圍內，也使被攻擊者能有餘力服務正常而使用者，不致完全被攻擊癱瘓。而如果留有攻擊過程中的攻擊封包記錄，網管人員也可以針對攻擊歷程而分析出攻擊者的位置，進而做出事後的處置(例如訴諸法律行動)。

我們針對於網際網路上會發生的 DoS/DDoS 攻擊來源追蹤進行研究，而研究的重點則是放在 TCP/IP 通訊協定中，對攻擊來源的 IP 位址進行追蹤，也就是希望能經由一程序來找出攻擊者真實的 IP 位址。由於 TCP/IP 通訊協定在制定之初，是考慮如何將封包由網路中一個節點傳送到另一個節點爲主的設計方式，因此並未投供封包來源 IP 位置的正確性認證，而有心的使用者就可以利用這個設計上的缺陷，假造攻擊封包的來源 IP 位址(IP spoofing)，這造成了追蹤上的困難。在研究上及實務上，追蹤到直正的攻擊來源 IP 位址不容易達成，但如果能追蹤到最接近攻擊來源的路由器(router)，也是一個可以接受的方案。藉由找出最接近

攻擊來源的路由器，就可經由阻斷，限流，和過濾等種種方式將攻擊對受害的影響減到最低。

我們研究了各種常見的 DoS/DDoS 攻擊方式及其運作原理，同時也熟悉一些實驗性攻擊工具的使用，對於現有的 IP 位址追蹤技術，我們也進行了深入的研究，同時，我們在實驗環境中實測了 SPIE 單一封包路徑備追蹤系統，並且自行實作 ICMP 訊息系統來進行封包採樣追蹤技術，在過程中，我們了解到各個技術的優缺點，並且了解到理論研究與實際運用上還有相當大的落差要克服，而我們也有另外整理 IP 位址追蹤實驗教材，包括”DDoS 攻擊追蹤報告”以及”SPIE 攻擊追蹤實例”，收錄在”NBEN 網路安全建置與實驗計畫--網路安全教育訓練教材”中。

而東華主要是針對高速網路下提高網路入侵偵測成功率技術做研究, 對 NIDS 有初步了解之後不難發現其執行效能的問題點在於對整個封包進行比對，往往比對的速度跟不上封包流經過的速度，造成在大流量之下有攻擊的情形發生卻無法百分之百作即時的反應，而導致NIDS 失去原有的功用。因此如何讓系統可以在高速中正常運行，有幾個層面可以改善和加強:

- 從比對演算法著手:

不論是以軟體或韌體來實作，都可透過改善演算法的方式加強偵測系統的效能。將比對規則前置處理之後，以DFA (Deterministic Finite Automata) 方式存在記憶體中，一定比用NFA要來得有效率。

- 由硬體設施下手:

CPU 速度或是IO 存取的速度，都有可能是瓶頸所在，這部份只能使用更精良的設備來克服，例如使用雙CPU 的架構或是採用RAID 的技術。

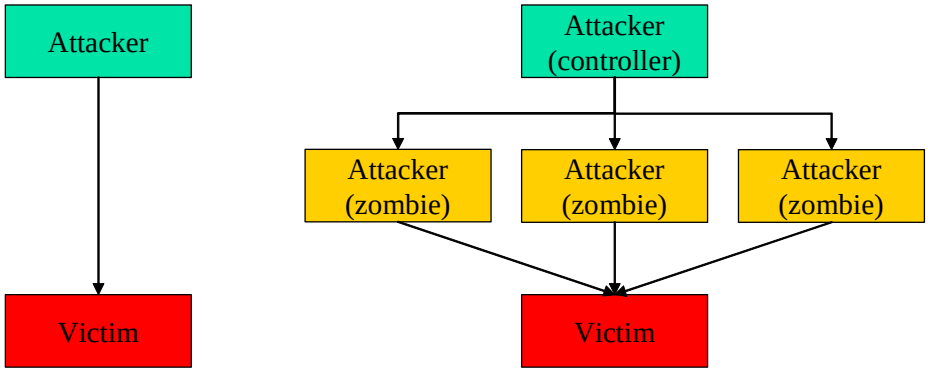
- 分流技術:

將大流量分成好幾條小流量到不同的偵測系統分別處理。分別從這三個層面下手，高速網路下的入侵偵測系統將成爲可行。

二、背景

DoS與DDoS攻擊

服務阻斷攻擊(DoS attacks, Denial of Service attacks)以及分散式服務阻斷攻擊(DDoS attacks, Distributed Denial of Service attacks)的原理很簡單，就是利用對攻擊的目標傳送大量的封包，使得被攻擊者的頻寬，記憶體，或是運算資源被消耗光，而無法或是很難提供正常的服務給正常的使用者，以達成服務阻斷的效果。DDoS 攻擊是 DoS 攻擊的延伸，其架構如下圖所示:



DoS　　DDoS

由圖中可以發現，DDoS 是 DoS 攻擊的進階版，DDoS 攻擊中的真正攻擊者退居第二線，控制數個被控制的攻擊者(Zombie attackers)，對目標進行攻擊，這樣的做法有兩個好處:

- DDoS 會比 DoS 攻擊更有效，主要是因爲流量加成(Traffic aggregation)的效果。由於 DDoS 攻擊的流量通常是由爲數很多的(被控制)攻擊者從各個不同的網域發動攻擊，這些不同來源的攻擊封包最後會被送到攻擊的目標，因此會有加成的效果。
- DDoS 比 DoS 更難處理與追蹤。DDoS 攻擊的來源數目很大，而每一個攻擊點並不用產生太大量的流量就可以有很好的攻擊效果，因此，發生攻擊時，網管人員通常無法在很短的時間內有效阻止攻擊，而在追蹤上，也造成了很大的困難，主要是因爲攻擊來源過多且過於分散。

以下是常見的 TCP/IP-based DoS/DDoS 攻擊，這種攻擊都是利用 TCP/IP 通訊協定設計上的缺陷來達成攻擊的目的。

TCP SYN-flooding：

攻擊者對其攻擊目標發送一連串的 SYN 封包，根據 TCP 通訊協定，被攻擊的系統會回應一個 SYN-ACK 封包，並且等待對方送出 ACK 封包以完成建立連線的程序，如果一段時間後還未收到 ACK 封包，則系統貯列中這個連線就會逾時時間而移除。但是由於短時間內有很多的連線要求，而攻擊者又不回應 ACK 封

包，最後被攻擊者系統貯列會因爲充滿了等待 ACK 的連線而造成無法再處理其他使用者的要求。

ICMP ECHO：

此攻擊是送大封包的 ICMP ECHO 給受害者，因 ICMP 的機制 受害者必需回同樣大小的 ICMP REPLY 到原送者，在這個同時，受害者不管是上傳或是下載頻寬都有等量封包進出,只要上或下達到百分之九十五的流量,攻擊就可達到效果。

ICMP SMURF:

此攻擊是將封包的來源 IP 位址設成攻擊目標的擴播 IP 位址，當受害者收到封包時會發送 ICMP REPLY 到自己的擴播 IP 位址，此時受害者的網路就會產生大量的封包碰撞，而使得被攻擊者的網路癱瘓而無法服務正常的使用者。

UDP:

此攻擊是送 UDP 的封包到沒有任何網路服務的 port 上 ，此時依據 UDP 協定，受害者有責任回 ICMP UNREACHABLE 的封包，這樣就可以達到攻擊的效果。

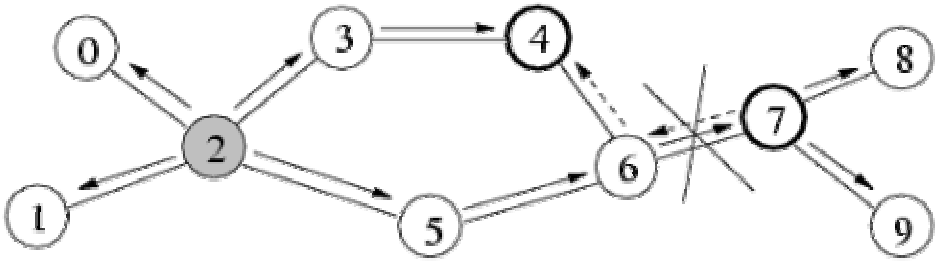
三、研究方法

I. IP 追蹤技術

台大的研究重點放在 TCP/IP 網路的追蹤的相關技術，主要目的是經由攻擊封包中的資訊，找出攻擊者的位置或其所在的大約位置。追蹤的問題之所以困難，主要是因爲基礎的 IP 協定中的封包來源是由發送者自己提供，並且沒有提供相對可能必要的認證資訊，所以讓攻擊者可以很容易地僞造大量來源不實的資訊來混淆追蹤者。針對這些特性，有不少相關方法被提出來解決這個問題。這些方法將在下面作一簡短說明。

入口過濾法(Ingress Filtering)

攻擊的封包常出現不實的來源位址，不正確的位址使得追蹤者無法得知攻擊者所在的位置。入口過濾法就是其中一種可以用來解決這個問題的方法，這種作法最主要是利用路由器來過濾掉來源位址不正確的封包。



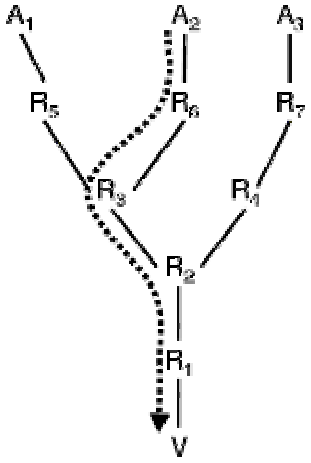
如上圖中所顯示的，各路由器都以數字標示，如果攻擊者的目標是位於 4 號路由器，而攻擊者從 7 號路由器傳送封包，而它所僞裝的位置是經由路由器 2 才可以達到路由器 4。很明顯地，如果路由器 7 有足夠的資訊，它可以判斷這個假造的封包不會由它傳到路由器 4(因爲路由器 2 傳到路由器 4 的封包不經過路由器 7)，因此它可以將之過濾掉。

這個辦法很簡單易懂，對於過濾資料來源不合法的封包也很有效，但是有一些缺點。首先，路由器通常只知道部分的網路路由器狀況，所以有時能作的過濾很有限，以之前的例子爲例，如果這個假造的封包的來源是屬於路由器 8，則無法被過濾，因爲路由器 8 的封包要經過它才能到路由器 4。一般來說，這個方法對 ISP 這類的系統最可行，不過，這些額外的需求，不只會增加路由器的運算量，也會增加管理的成本。這個方法對追蹤者來說有一個額外的好處，就是追蹤者可以把攻擊者的位置縮小到更精確的區域。

連結測試法(Link Testing)

這個方法的精神是，追蹤者先從最接近被攻擊者的路由器開始找起，確認攻擊封

包是由那一個路由器來的，確定了後，再往路由器的上一層找，重覆這個程序直到找到攻擊者爲止。這個方法最主要用在攻擊還持續在進行的時候。



如上圖所示，假設 A_2 是攻擊者，而 V 是被攻擊者，追蹤者首先確定攻擊封包是來自路由器 R_1，然後確定來自 R_2，R_2 之上有 R_3 及 R_4，而只有 R_3 有傳送攻擊的封包，依此類推，直到找到 A_2 爲止。而其中又有兩種不同的作法:

- **輸入除錯法(Input Debugging)**

有很多的路由器都含有輸入除錯的功能，這是指路由器管理者可以經由過濾出口的某些特定封包的入口是那裡，而這個功能可以用來追蹤。首先，被攻擊者要從各個攻擊封包中找出一些共有的特性，然後，將這些特性交給路由器管理者，管理者就可以用這個資訊來過濾到被攻擊者的出口的封包的入口(來源)在那裡。往上一直重覆這個步驟就可以找出攻擊的源頭，這類工作可由管理者或是自動化的程序來處理。

這個方法的最主要問題是，要有管理者的參與。而且事關數個路由器，就算有自動化的工具，也不能保證這些路由器的除錯工能會完全相容，這會造成相當大的負擔。

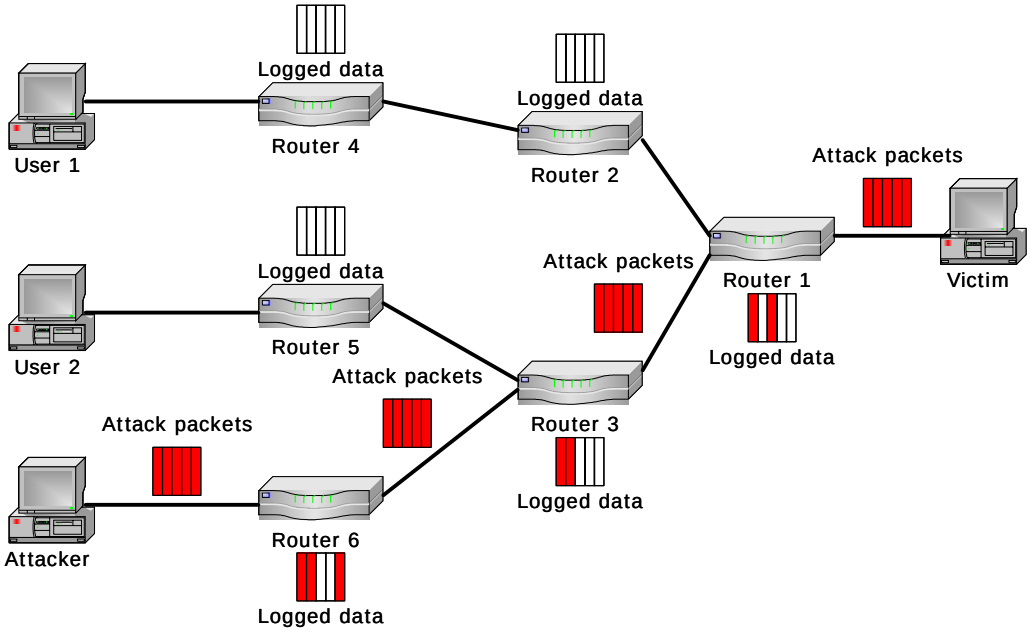
- **控制淹沒法(controlled flooding)**

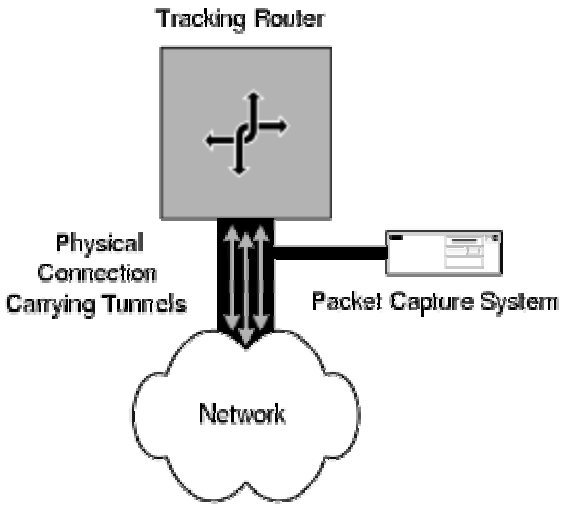
由於上一個方法在應用上需要路由器管理者的協助，所以針對這個缺點，另一個被用追蹤攻擊來源的方法被提出來。這個方法被稱爲控制淹沒法，是因它以傳輸大量的封包來對攻擊者封包造成的影響，來找出攻擊者的位置。其原理在於對一個路由器來說，緩衝區空間都是共用的，大量的封包會造成從攻擊者收到的封包有排擠的效果。而追蹤者可利用這些變化來猜出攻擊的所在。

不過這個方有一個主要的問題就是，這個方法會產生大量的封包，可能會被當成是一種攻擊，這對網路上其他不相干的機器來說是個負擔，而且被攻擊者必須對網路的路由器結構有很清楚的了解才能有效利用這個方法。對於分散式的攻擊者來說，這個方法也不適用，因爲不論對被攻擊者或網路本身來說，負荷都過大。

記錄法(Logging)

這個方法就是把各主要路由器之間的封包記錄起來，然後再用一些演算法來找出封包傳送的路徑:



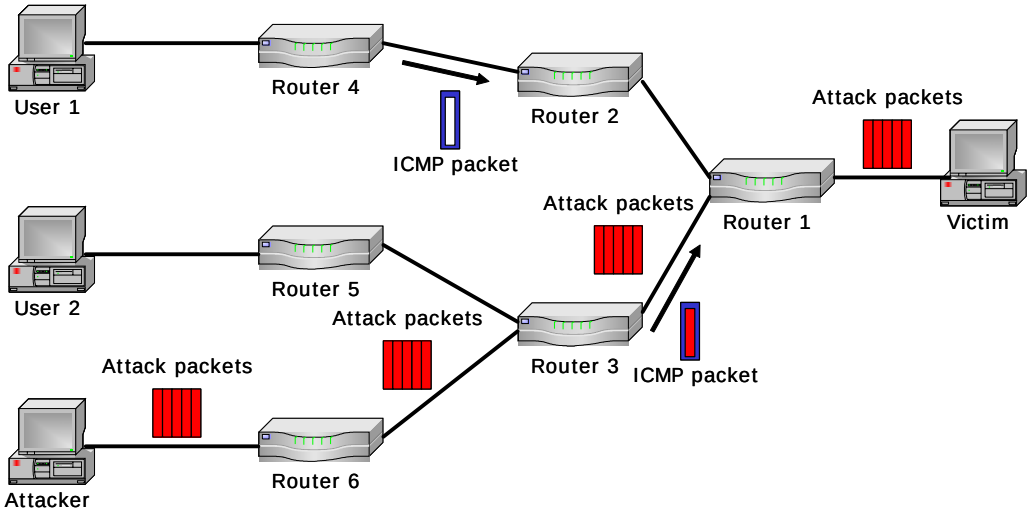


利用一個封包截取系統(PCS，Packet Capture System)的輔助，除了可以記錄以外，還可以做到包括除錯等更複雜的工作或分析，而且可以在路由器本身不支援的情形下也可運作。

這個方法的最大好處就是 PCS 是獨立於路由器之外的，換句話說就是不需要路由器提供的特殊協助，有了更大的彈性，而且分析記錄可以在攻擊停止之後。其缺點就是要負擔額外的設備，及可能需要的大量儲存裝置。

ICMP 回溯法(ICMP Traceback)

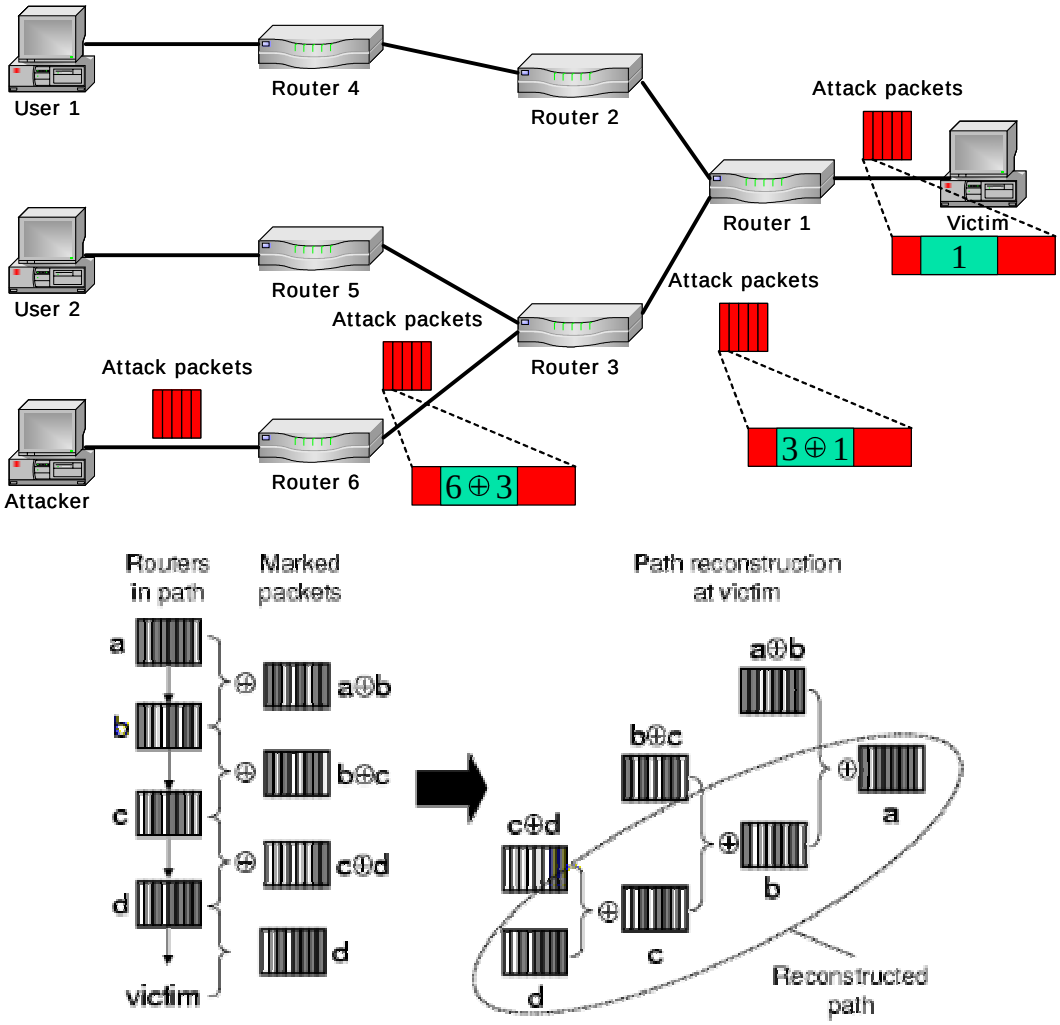
這個方法主要是由路由器另外產生特殊的 ICMP 回溯訊息，然後用這個訊息來做到追蹤的功能。這個想法是，有一個很低的機率(比如說 1/20000)，路由器會把封包傳送相連路徑上的資訊包在回溯訊息中送到目的地。在攻擊訊息出現時，被攻擊者可以收集這些資訊，並用來推斷出攻擊者所在位置。



這個方法要的運作方式，其實十分類似於記錄式系統，而其最大的不同點在於，訊息式系統不會把採樣的封包記錄下來，而是把它包裹在 ICMP 封包中，往採樣封包中指定的目的地傳送。這當然是這個方法最大的缺點有兩個，首先，這個方法會對網路頻寬有一定的影響，而且 ICMP 封包很可能因爲路由器或防火牆的設定而被過濾，再者，被攻擊者不一定有能力處理這些封包或進行追蹤，所以在缺乏配合下，就會造成浪費。這個方法的好處是，不需要大量的儲存裝置，而且，ICMP 封包中可以包含任意的追蹤資訊，可以讓追蹤上更爲容易。

封包標記法(Packet Marking)

爲了將路由器的負擔減到最小，利用現有的封包的表頭來夾帶一些可用來追蹤的資訊的方法就是封包標記法。這個方法主要分成兩個部分，標記演算法以及路徑重組演算法。標記演算法是由路由器來執行，用來將資料加入現有的封包表頭中，包括路徑中位置的 xor 值，及用來檢查的雜湊值，而追蹤者收集這些有被標記的封包，利用其中的資訊來推算出封包的來源位置。



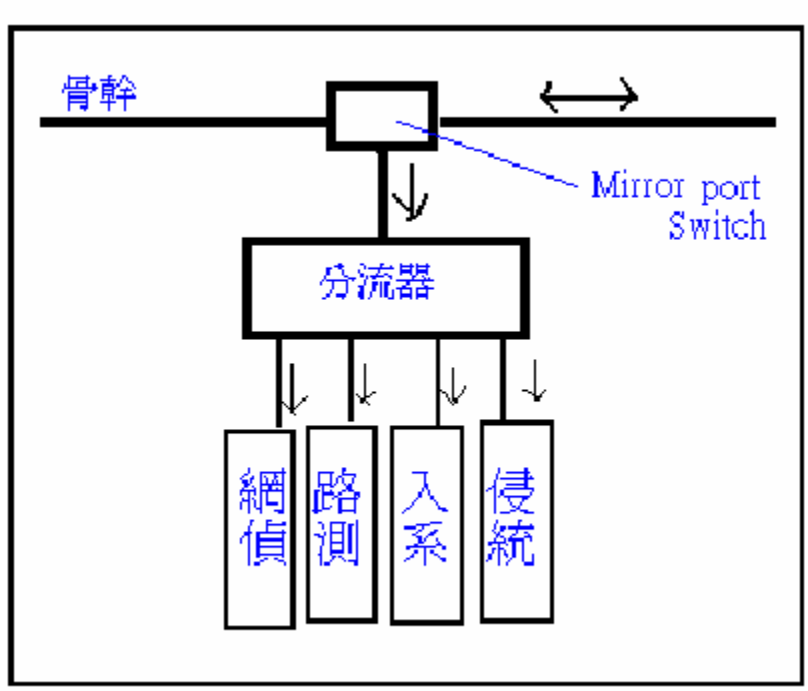
如圖中所示，使用 xor 將路徑中相鄰的位置值作運算，在路行重組時再將整個的路徑算出是一個有效結省封包位元數的方法，可將額外資訊的增加量減到最低。這個方法的最大好處就是不用額外的封包來傳遞追蹤訊息，不過缺點是用來存放 xor 值的 ID 欄位 IPv6 不支援，且某些其他協定會使用到，會造成運作不正常。

這個方法的最大優點是不會增加網路的流量，因爲所有的資訊都已包含在封包中。但是它確有很大的缺點，使得這個方法在實用上有其局限性，首先，這個方法需要路由器的配合，而在現實的網路環境中無法做到，另外，這個方法只適合 IPv4 的協定，對於其他的包括 IPv6 或 IPsec 等未來可能被採用的協定都無法使用，而就 IPv4 封包而言，或標頭欄位中只有 16 位元是可以用來夾帶追蹤的資訊，這使得追蹤者必需要取得一定數量的封包才能得到足夠的資訊，如此也增加了追蹤上的困難。

II. 網路分流技術

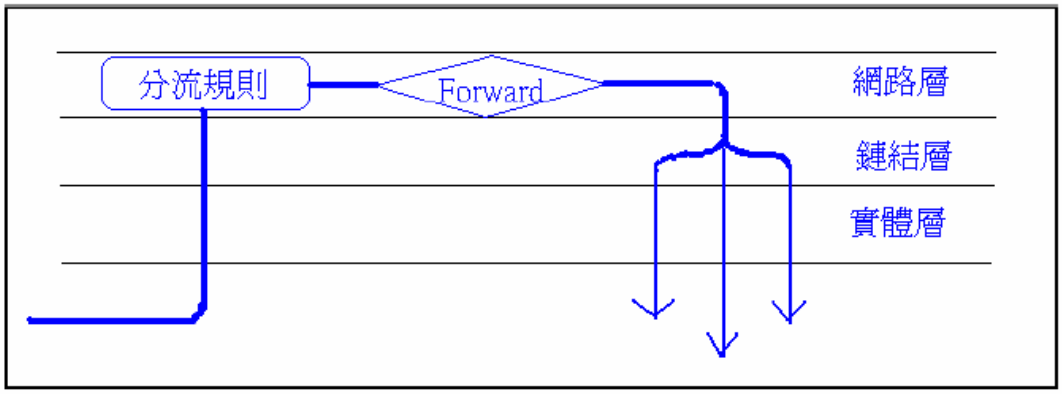
東華的研究重點放在網路的分流技術。分流的概念很直覺，悉將由骨幹中複製一

份封包到分流器中，經處理後成較小流量後分別轉送到不同的偵測系統進去分析的工作。很清楚的可以看到分流的工作是由分流器所負責，一般的做法分爲兩種，第一個選擇是從網路層(OSI第三層)下手，另一個選擇是由連結層著手(OSI第二層)。



網路層下的分析

首先，先了解如何網路層作到分流的功能，想法很單純，只要將流入分流器封包的目的IP 位址改成偵測系統的IP 位置，此時分流器就當起路由器的功能會自動將封包送往NIDS，而剩下的工作則是攥寫分流規則，辨定進入的封包要由那一個介面出去，例如目的port80 送往第一台NIDS，而目的port 23 送往第二台。

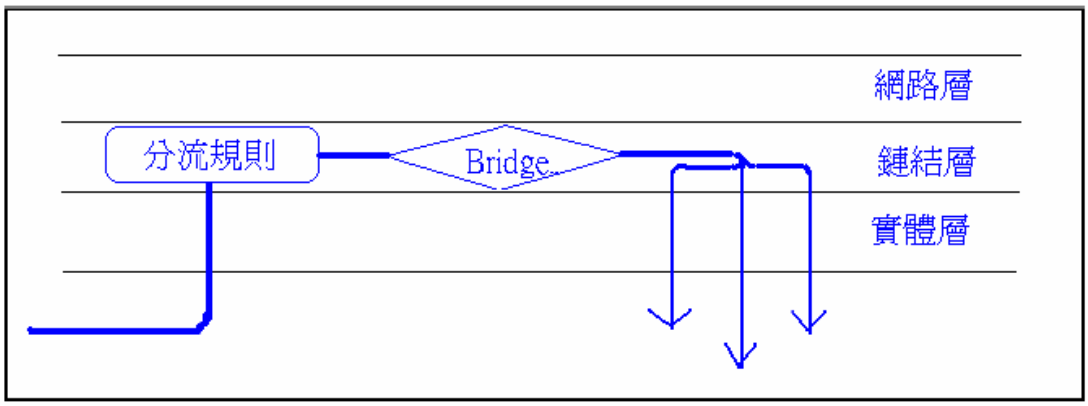


這種做法有的優點是，它功能強大，而且分流規則可以依IP、ICMP、TCP 或

UDP 標頭中的欄位來指定將封包由不同介面送出去。它也可以支援round-robin的功能在使用iptables(Appendix B)的環境下，可以針對同一種攻擊類型round-robin到不同的偵測系統，以便平衡負載，當骨幹中同一種攻擊有很大的比例時將很有用處。而它的缺點是目的IP位置被改變因透過改變目的位置，來達到分流的功能，在攻擊紀錄檔上的辨讀上，目的IP將無參考的價值。

鏈結層下的分析

在了解分流在網路層下的作法之後，接下來分析在鏈結層下分流的方法，其實跟在網路層下大同小異，只要將進入封包的目的MAC修改成偵測系統的的MAC位置，剩下的分流器會以橋接的方式直接送過去。



這個方法的優點在於網路層以上的資訊得以完整保留，整個過程中，只會變動到MAC位置，不會影響到攻擊紀錄的辨讀。而且它的效能也比網路層的系統高，因爲只需到第二層就可以決定要橋接到那個介面去，如果在高速中換成是分流器遇上瓶頸，即可採用此種方法。不過缺點是其功能有限就目前ebtables 的功能而言，只要針對IP 標頭中少數的欄位作比對以決定要橋接到那個介面，這對分流器來講是不夠用。

橋接器的必要性

然而這兩種方法看起來都行得通，但是別忘了一點重要的網路觀念，就是從骨幹複製過來封包的的MAC 並不是分流器的位置，這樣會導致封包在OSI 七層中的第一層就被丟棄，根本無法往上層送，這時候橋接器剛好可以派上用場，因此來了解一下橋接器是如何工作的:

1. 如果目的MAC 位置在橋接器的另一端，則直接橋接過去。
2. 如果橋接器不知道目的MAC 位置，則送往每個介面。
3. 如果目的MAC 位置是橋接器本身，則往OSI 上層送去。
4. 如果目的MAC 位置在橋接器的同一邊，則不管他。

因此不管是使用第二層或第三層來分流，只要一直把橋接器的功能原理放在腦海裡，發生問題時將會更容易解決。

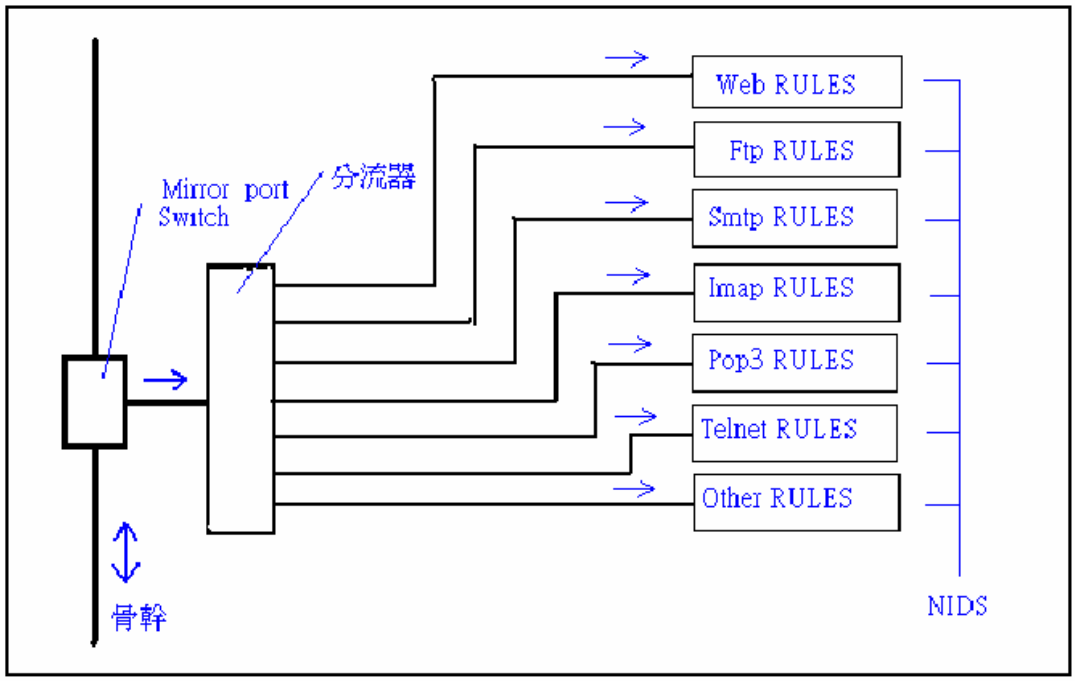
分流規則的研究

- **靜態的負載平衡**

每一台NIDS 載入相同的比對規則，而分流器將流入的封包採依序等量的分法。其優點每一台都公平地分配到封包，不會產生有的系統CPU 負重很高，而有的卻閒置，而且這個方法實作簡單分流規則明確且不需要考慮封包中其他欄位，對封包一視同仁依序分配。不過缺點是對整體效能提升有限，由於所載入的比對規則並沒有減少，每一台NIDS 每單位時間內所能處理的流量還是一樣，因此再一度發生瓶頸時只能加入更多主機來分擔流量。而且無法偵測有狀態的攻擊有的攻擊並不是在一個封包之內就可以完成，然而靜態負載平衡會將有關係的封包分配到不同的NIDS，會導致失去偵測這種類型攻擊的能力。

- **分類分流導向**

每一台NIDS 依封包特徵(例如:web 的攻擊大多爲Port 80)載入不同的比對規則，而分流器將流入封包依指定特徵分到不同的NIDS。



它能讓效能大幅提升，主要是因爲分流器分類方式來分流，在偵測系統的觀點來看，不僅僅是流量變少，連載入的比對規則也大幅度的減少，更能因應高速下的嚴苛考驗。這類系統可以更進一步做到可偵測部份有態攻擊假設有態攻擊的所有封包中的某個特徵都相同(例如:目的port)，而分流也是依照這個特徵來決定由那個NIDS 負責，將可以被偵測出來。它的缺點是分配上會造成不公，有可能在同一時間內，某一種攻擊類型大增，導致其他NIDS 閒置，而只有一台處於非常忙

碌的狀態，而且分流器工作量也較重，依類型來分流，意謂要爲每個類型加上分流規則，而當封包進入分流器時則需依序通過每一條分條規則，等到符合條件成立才轉送到其他介面，而規則一多工作量相對變大。

- **動態負載平衡**

除依封包特徵之外，加入動態支援的功能，當其中一台CPU 負載過重時，會主動通知分流器和其他NIDS，經過溝通之後選擇一台閒置的NIDS 切換所載入的比對規則而同時間分流器也改變分流規則，將原來前往負載過重主機的封包，分一半到切換比對規則的主機，而原來流向切換比對規則主機的封包改流到一台載入所有比對規則的主機。

優點是提高對環境適應力，但相對地，這也是最複雜的一種，目前在分流器和各偵測系統之間並沒有安全又簡單的解決方案。自行定義新的通訊協定可能是比較好的解法，而且對有態攻擊失去偵測能力。

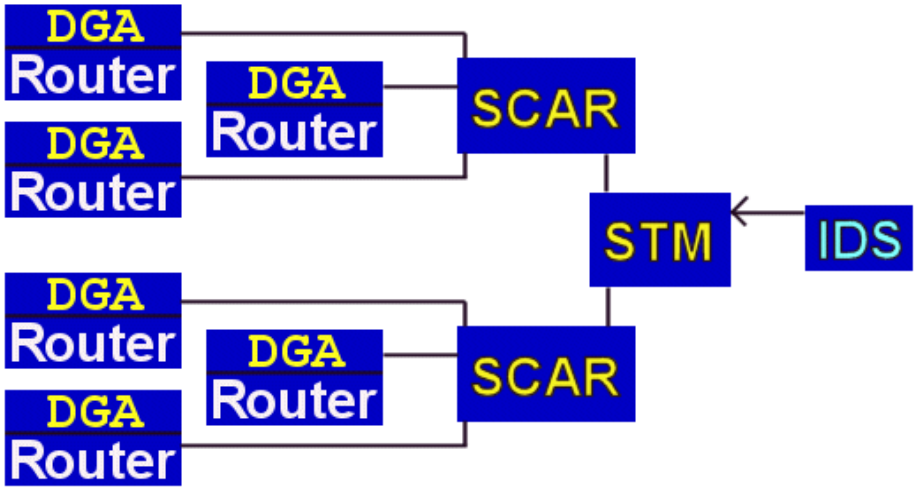
四、研究成果

I. IP 追蹤技術

本計畫希望能針對可行的 IP 位址追蹤技術進行研究，由目前各項相關技術中，我們選擇記錄式(logging)系統與 ICMP 訊息式(ICMP messaging)系統進行研究，這兩種方法的主要優點是不需要對路由器進行客制化，因此在實際應用上會比封包標記法要好，而輸入除錯法和控制淹沒法則通常需要啓動路由器的服務或是對路由器的設定進行改變，因此彈性上較少。

SPIE(Single Packet Isolation Engine)記錄式追蹤系統

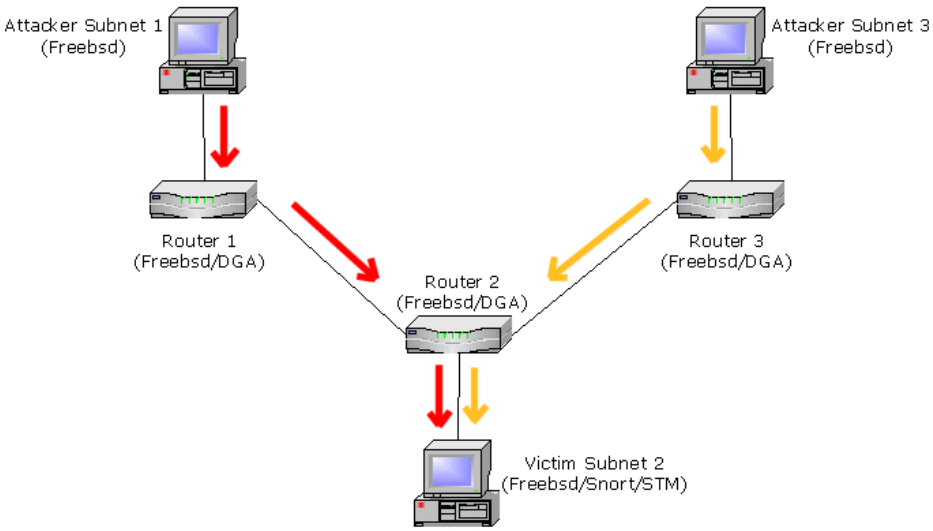
BBN Technologies 提供了一個可供識別單一獨立封包來源的工具：SPIE(Source Path Isolation Engine)。SPIE 記錄了每一個通過路由器的封包，當需要追蹤某封包傳遞路徑的時候，只需要返向詢問每一台路由器是否曾經看過這個封包即可，藉著這個過程，即可逐步重建出封包的傳遞路徑，甚至找到最接近攻擊者的網域或路由器。SPIE 可以針對任何單一封包進行追蹤，利用建置在各路由器上的封包記錄，SPIE 可以重建出封包流過的路徑，由此我們可以得知封包來源。



SPIE 的架構如上圖所示，首先透過 IDS 鑑別出攻擊封包，通知 STM 並送出相關訊息，如封包內容、到達時間等，由 STM 要求 SCARs 去收集各 DGA 之前儲存的封包摘要表格，和製作摘要的雜湊函式、時間區間等資料，找出那些路由器曾經送過這個封包，並重建起封包走過的路徑，各 SCAR 再將區域性的路徑交給 STM 重組成完整的封包傳遞路徑。

下圖說明一個 SPIE 的追蹤實例，將由兩台 Attacker 分別由不同網域發出單一攻擊封包，目標是 Victim。首先，第一回合(紅色箭頭所示)，Attacker 1 會送出第一個攻擊封包，此時 Attacker 2 所在的網域可以發出一些普通的封包到其他網域，而唯一的攻擊封包會從 Attacker 1 經過 Router 1 和 Router 2，抵達 Victim，而 Victim 必須在來自各地的封包中發現這個唯一的攻擊封包，並且透過 SPIE 找

出此攻擊封包經過的路徑，結果必須是 Router 1 -> Router 2，而非 Router 3 -> Router 2 或其他結果。這回合完成之後，第二回合(橘色箭頭所示)再從 Attacker 2 送出另一個單一的攻擊封包，此時 Attacker 1 將不發出任何攻擊封包，改送出普通的封包到其他網域，同樣地，Victim 必須在 network traffic 中發現 Attacker 2 的攻擊，並且找出 Router 3 -> Router 2 -> Victim 的路徑。



在 Attacker 1 向 Victim 發動攻擊後，Victim 上的 Snort 會偵測到攻擊，並通知 STM，STM 會先發出警告(***開頭的訊息)，然後等到 SCARs(或直接由直屬的 DGAs)回報追蹤結果，將部分的追蹤路徑組織好之後，就會出現 STM 已完成 traceback 行動，管理者可以閱讀最終追蹤結果的訊息(---開頭的訊息)

```
spie> *** ATTACK DETECTED: type "s 48104ef7" for more information.
--- answer received: type "s 48104ef7" for more information.
```

而追蹤的結果會得到一個類似下圖的訊息

```
spie> s 48104ef7
Trace ID: 48104ef7
Active: false
TRACE REQUEST     message=0x48104ef7     initial     pktlen=48     1 queries
  PACKET:   45 00 00 30   41 7a 00 00   03 11 9d 94   01 02 03 04
            c0 a8 14 01   54 aa 30 39   00 1c 00 00   00 00 00 00
            00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
  QUERY 1     0xc0a814fe     <unbounded,13:19:37.745433>
TRACE REPLY     message=0x48104ef7     0 continuations     end-of-reply
  NEIGHBOR 0     spieid=0xc0a80afe   latency = (0 - 0) usecs
  NEIGHBOR 1     spieid=0xc0a81efe   latency = (0 - 0) usecs
```

```
  NEIGHBOR 2      spieid=0xc0a814fe    latency = (0 - 0) usecs
  RESPONSE 1      dga=0xc0a814fe      pkt=0
     TRACE 1 pos bound=13:19:37.750433
<13:19:28.946812,13:19:38.946942>
         NEIGHBORS:         0         1
  RESPONSE 2      dga=0xc0a80afe      pkt=0
     TRACE 1 pos bound=13:19:37.750433
<13:19:32.973776,13:19:42.973934>
         NEIGHBORS:         2
```

我們可以從追蹤報告中看到，這次追蹤報告的代碼(48104ef7)，封包的完整內容(以 HEX 表示)，和這次查詢的內容，包括以上的 packet 內容，最靠近被攻擊者(victim)的路由器的 SPIE ID (此例爲 0xc0a814fe)，還有此封包抵達被攻擊者時的時間(此例爲 13:19:37.745433)。接著就是追蹤報告，此例中收到兩個 DGA 的回應(Response)，分別是 dga=0xc0a814fe(DGA-2)和 dga=0xc0a80afe(DGA-1)，後面則是該封包抵達該 router 的時間區間(此例爲 10 秒)，若在此時間區間內在該 DGA 上找到此封包的雜湊值，DGA 就會回報此封包曾經通過此路由器，依據時間先後及路由器的實體連結情況，我們就可以得知該攻擊封包曾經通過 DGA-1 和 DGA-2。

ICMP 訊息系統與流量偵測

由於攻擊程式通常會送出數倍於正常使用的網路流量，因此我們觀察經過特定網路節點(路由器、閘道器、主機等)的封包，並在某一特定機率下，選取當時流經的封包，紀錄其資訊，並送出一個特製的 ICMP 封包至該流經封包的目的地。

當攻擊發生時，流經相同網路節點的攻擊流量通常會數倍於正常的流量，因此會有較多的封包資訊被紀錄並送至受害端(victim)。受害端根據這些搜集到的資料加以統計，可粗略判斷出真正發出攻擊封包的網路位址。

爲保證送出的封包資訊的不被僞造，我們使用單向雜湊函數(MD5)對事先設定好的共用密碼以及封包資訊做運算後，將其結果連同封包資訊一同傳送。產生的封包資訊格式如下(ICMP 封包)：

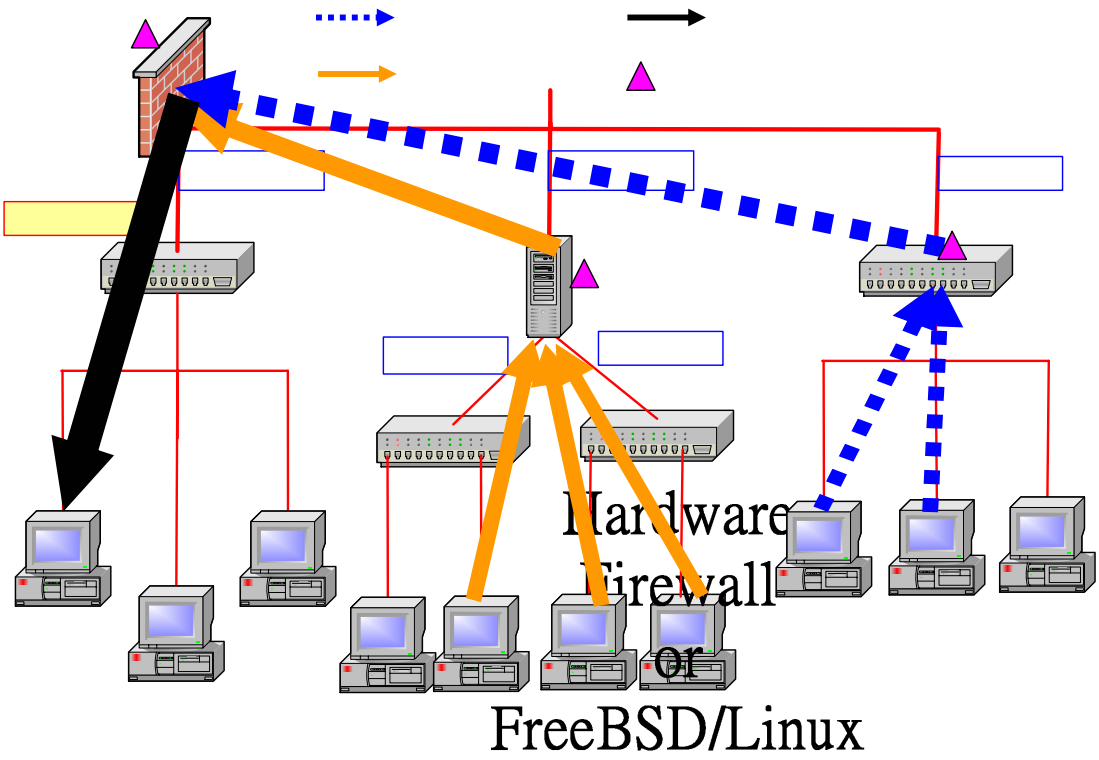
0	8	16	24	31
Type	Code	Checksum		
Un-used				
Stamper ID				
Source IP				
Destination IP				
Timestamp (part of seconds)				
Timestamp (part of microseconds)				
MD5 Checksum (byte 1-4)				
MD5 Checksum (byte 5-8)				
MD5 Checksum (byte 8-12)				
MD5 Checksum (byte 12-16)				

每個送出 ICMP 追蹤封包的主機需要將所有流經的封包資訊記住，以提供往後受害端查詢封包之真正來源。我們使用 Bloom Filter 粗略的記住流經的封包資訊，以節省記憶體的空間。以長度爲 28-bit 的 Bloom Filter 雜湊空間約需 32MB 的記憶體。當受害者搜集足夠的 ICMP 追蹤封包後，從較可疑的網路(送出大量異常流量)開始查詢，確認攻擊封包的真正來源。

首先，我們對發送訊息對網路頻寬造成的影響進行評估，在不同的產生機率下，ICMP 追蹤封包對正常流量的比值及流量(理論值, 以 10M 資料量來計算)，如下表所示:

產生機率 封包大小	1	$\frac{1}{10}$	$\frac{1}{100}$	$\frac{1}{1000}$
64	1 (10M)	$\frac{1}{10}$ (1M)	$\frac{1}{100}$ (102K)	$\frac{1}{1000}$ (10K)
512	$\frac{1}{8}$ (1.25M)	$\frac{1}{80}$ (125K)	$\frac{1}{800}$ (12.5K)	$\frac{1}{8000}$ (1.25K)
1498	$\frac{1}{23}$ (427K)	$\frac{1}{234}$ (42.7K)	$\frac{1}{2340}$ (4.3K)	$\frac{1}{23406}$ (0.4K)

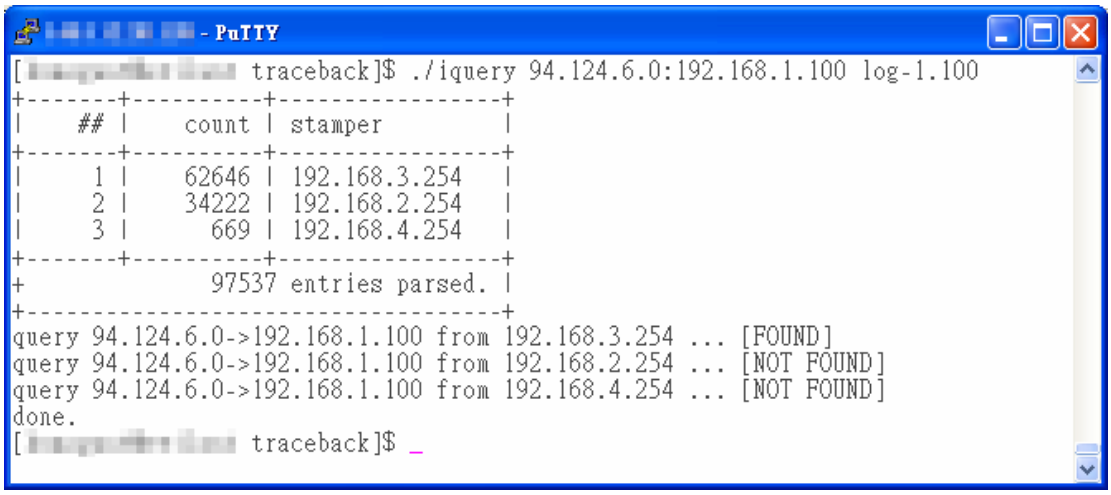
在我們實作的訊息系統中，我們只針對邊緣路由器(Edge routers)所流經的封包進行採樣，這樣作的目的最主要的根據以下的假設，攻擊者和被攻擊者都位於邊緣路由器所在的網域中，如果我們可以在網際網路上各個邊緣的路由器都建置追蹤的系統，那即使非邊緣伺服器都沒有支援追蹤機制，也依然可以達成追蹤攻擊來源的目的。



我們從上圖中可以看到此系統的建置方式，而我們在邊緣伺服器上建置訊息採樣與 ICMP 追蹤訊息發送 daemon，並在受害者端先計來自特定邊緣伺服器的訊息，下表是觀察真實發送的網路，由 ICMP 封包的資料，受害端可整理出下表的紀錄：

##	count	Stamper
1	62646	192.168.3.254
2	34222	192.168.2.254
3	669	192.168.4.254

可以看出有攻擊封包流經的節點其統計數量明顯的比正常流量的大。透過我們的查詢程式，我們可以查到某個特定位址 92.124.6.0 是從 192.168.3.254 該網路傳送過來，如下圖所示：



這個系統最主要可以發現不正常的流量，而且由於 ICMP 追蹤訊息都有經過簽章，所以我們可以保證我們所收到的追蹤訊息確實由我們所建置的追蹤器所發

出，而非由攻擊者所爲造的資訊。

II. 網路分流技術

測試工具說明

在測試中，會使用到二支測試工具，第一支是自行攥寫的程式transp，用於產生異常封包。第二支是網頁伺服器的測試工具ab，用於模擬現實生活中的正常封包。在這裡簡單說明使用方法。

Usage : ./transp [--option1] [-option2]
option1:
--sport arg : specify the source port
--dport arg : specify the destination port
--seq arg : specify the sequence field in TCP header
--ack arg : specify the acknowledge field in TCP header and enable the
ack flag
--urg : enable the urg flag in TCP header
--psh : enable the psh flag in TCP header
--rst : enable the rst flag in TCP header
--syn : enable the syn flag in TCP header
--fin : enable the fin flag in TCP header
--help : you will see this
option2:
-c arg : how many packet do you want to transit
-d arg : specify the destionation IP field in IP header
-i arg : specify the identificatioin field in IP header
-s arg : specify the source IP field in IP header
-p arg : specify the payload length after IP header
-q [T|R|C] : specify the tos field in IP header
T : stands for max throughput
R : stands for max relibility
C : stands for min cost
-t arg : specify the ttl field in IP header
-v arg : specify the interval of each packet

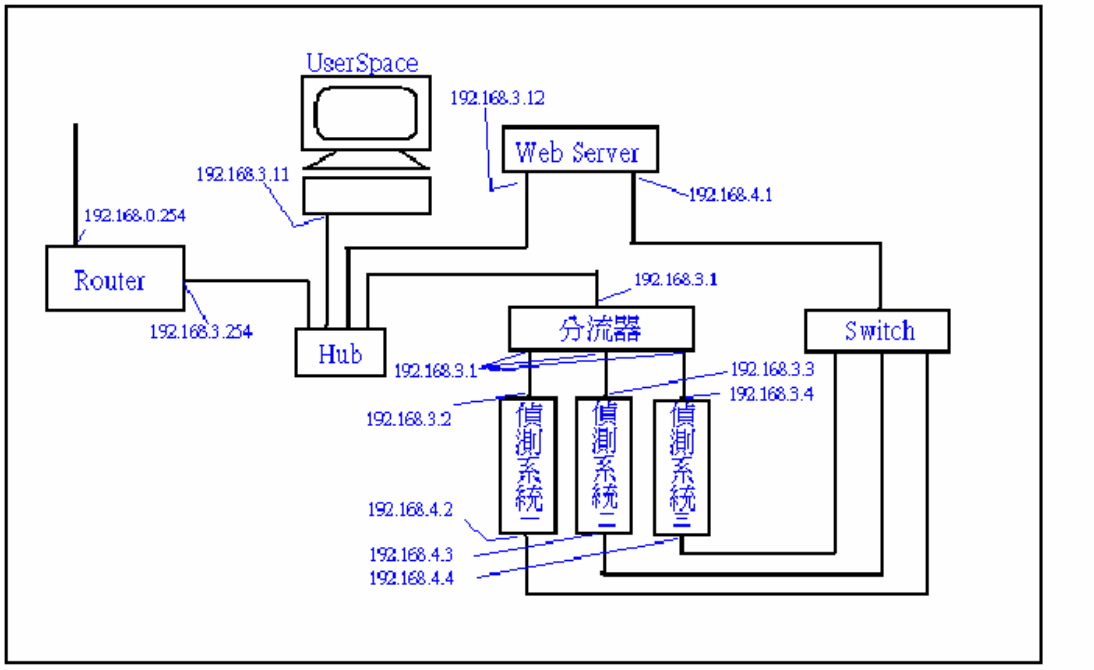
以上的參數可以混合使用，而在沒有指定的情況下，將以亂數的方式產生。另一個工具的使用方法:

Usage: ./ab [options] [http://]hostname[:port]/path
Options are:
-n requests Number of requests to perform

環境與測試方法

接下來，將在實驗的環境下模擬攻擊，測試的重點有下列幾項:

- 載入比對規則多寡對偵測系統的影響。
- 是否可成功分流，在各偵測系統上達到負載平衡參測試有無分流對偵測系統負載的影響。



測試的流程如下

1. 分別由192.168.4.0 網域的介面登入各偵測系統，執行top 指令並設定間格爲一秒重整，以方便往後CPU 負載的紀錄。
2. 在Router 上安裝測試工具，並以來源IP 爲192.168.0.254 目的IP 爲192.168.3.12 的方式，對Web server 作出不正常封包的攻擊。
3. 封包在Hub 中散開，因此除了Web server 之外，分流器也會接收到不正常的封包。
4. 當不正常的封包進入分流器時，將依照事先所給定的分流規則作出分流的動作並向後傳給偵測系統。
5. 偵測系統接收由分流器所送過來的不正常封包，經比對規則辨定是不正常的封包之後，經由偵測主機的另一個於192.168.4.0 網域的介面，將攻擊紀錄傳送到資料庫，同時間紀錄CPU 負載的變化。
6. 管理員於UserSpace 的機器前面，透過瀏覽器瀏覽存於Web Server 資料庫中的攻擊紀錄，辨讀是否有達到所要測試項目。
7. 重複1到7的步驟，分別作不一樣的測試。

測試結果

- 測試一: 比對規則多寡對偵系統負載的影響
  - ◎ 測試指令./ab －n 290 192.168.3.12/index.html對Web server 中的 ndex.html 作290 次的要求。
  - ◎ 比對規則偵測系統一載入比對所有目的port 爲80 的規則。
  - ◎ 分流規則所有封包給轉送到偵測系統一。

測試得到的CPU 使用率如下:

第一	7	19	55	52	63	45	56	53	57	50	40	7
第二	7	27	55	50	53	51	56	54	56	58	45	7
第三	7	51	59	52	46	57	47	50	51	55	30	7
總和	21	97	169	154	162	153	159	157	164	163	115	21
平均	7	32	56	55	54	51	53	52	55	54	38	7

記憶體使用量均爲3308Kbyte。

  - ◎ 測試指令./ab －n 290 192.168.3.12/index.html。
  - ◎ 比對規則偵測系統一載入比對目的port 爲80 的部份規則。
  - ◎ 分流規則所有封包轉送到偵測系統一。

測試得到的CPU 使用率如下:

第一	7	23	32	32	34	34	33	42	30	29	36	7
第二	7	37	36	34	32	28	31	27	38	33	27	7
第三	7	27	39	35	40	38	34	36	36	39	28	7
總和	21	87	107	101	106	100	98	105	104	101	101	21
平均	7	29	36	34	35	33	33	35	35	34	34	7

記憶體使用量均爲1304Kbyte

由上兩表可以得知在相同流量下載入愈少的比對規則，CPU 的使用量也會跟著變少，而分類分流目得正是要減少載入規則數目。

- 測試二: 分流對偵測系統負載的影響
  - ◎ 測試指令./transp--syn --seq 2 --dport 80 -d 192.168.3.12 -c 500 -p960。
  - ◎ 將送500 個長度爲1000 bytes 、sequence 爲2 、目的port 爲80、目的IP

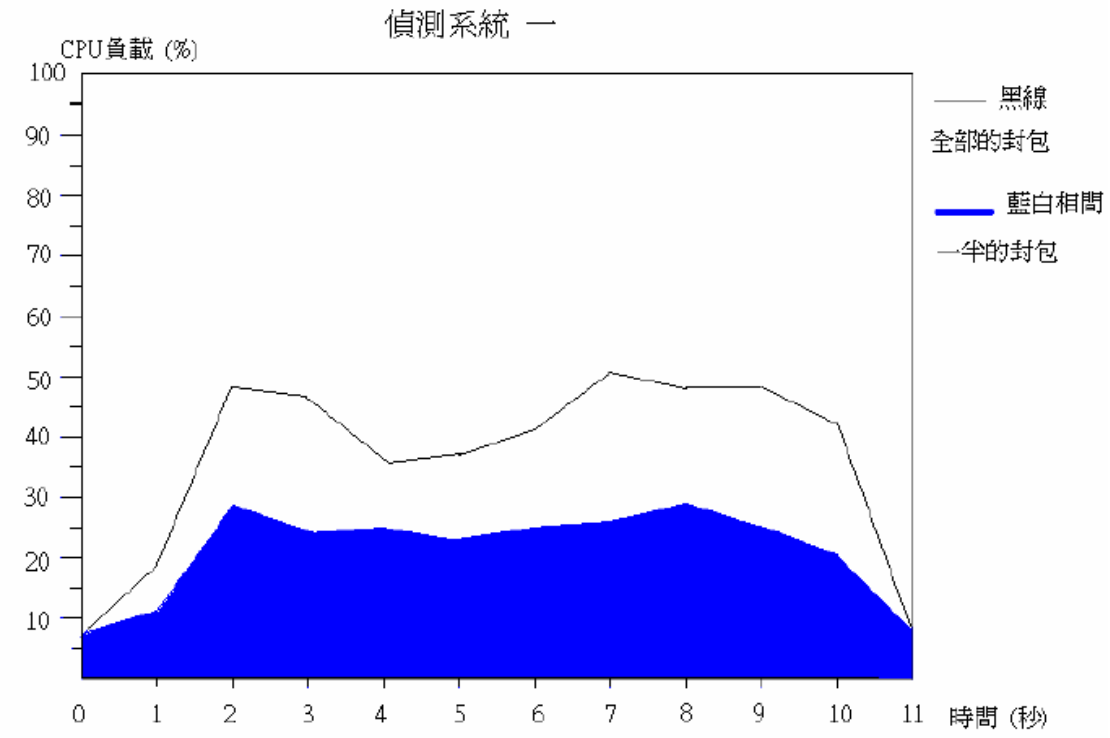
爲192.168.3.12 的封包，且在第十秒測試工具會自動結束。

◎ 比對規則偵測系統一載入比對異常封包的規則，所有封包轉送到偵測系統一。

測得的CPU 使用率如下:

第一	7	28	54	55	51	28	46	53	40	42	52	7
第二	7	17	55	38	37	54	40	44	54	55	52	7
第三	7	12	35	44	21	29	38	55	52	50	22	7
總和	21	57	144	137	109	111	124	152	146	147	126	21
平均	7	19	48	46	36	37	41	51	49	49	42	7

三次的攻擊紀錄檔，Alert 次數均爲500 Lose 次數均爲0將數據繪圖如下(黑線的部份代表CPU 使用率的變化)。



◎ 測試指令./transp--syn --seq 2 --dport 80 -d 192.168.3.12 -c 500 -p960。

◎ 比對規則偵測系統一和偵測系統二載入比對異常封包規則。

◎ 分流規則平均分給偵測系統一和偵測系統二。

測得到的CPU 使用率如下:

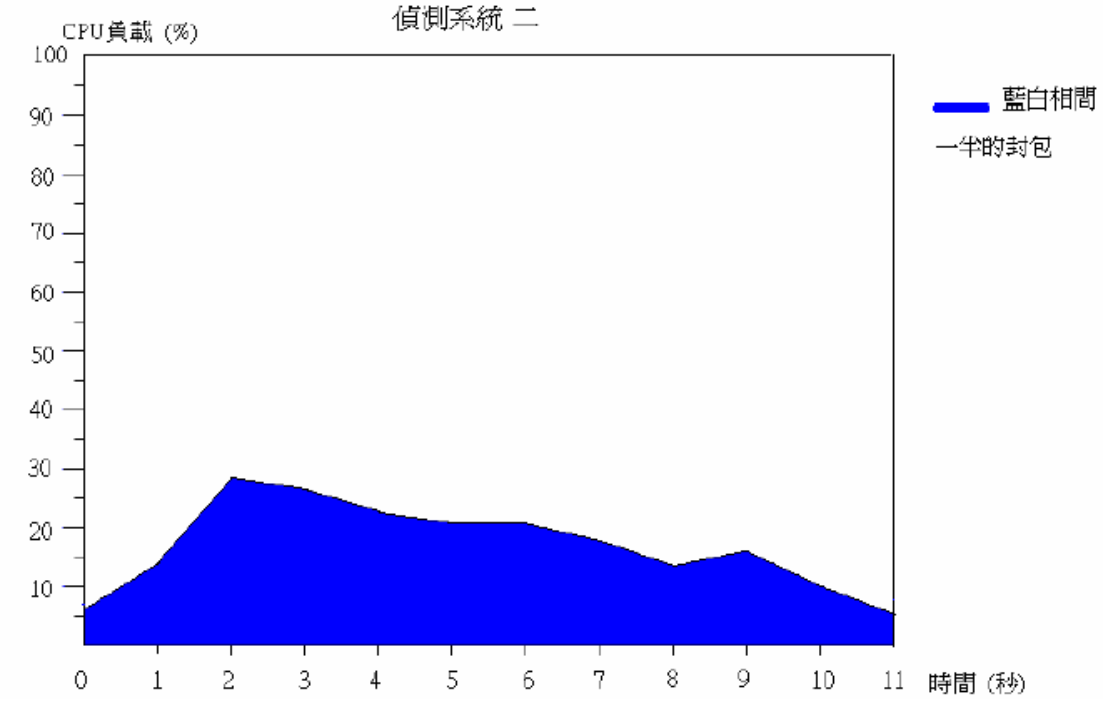
偵測系統一

第一	7	13	26	27	16	16	29	30	31	25	16	7
第二	7	9	29	13	29	22	17	24	30	29	30	7
第三	7	11	31	31	29	30	29	23	17	20	16	7
總和	21	33	86	71	74	68	75	77	88	74	62	21
平均	7	11	29	24	25	23	25	26	29	25	21	7

偵測系統二

第一	6	10	28	29	27	17	9	7	8	9	9	6
第二	6	19	29	23	18	20	28	25	20	29	24	6
第三	6	13	29	25	24	26	27	23	13	11	10	6
總和	18	42	86	77	69	63	64	55	41	49	43	18
平均	6	14	29	26	23	21	21	18	14	16	14	6

以上偵測系統一和偵測系統二，三次的攻擊紀錄檔，Alert 次數和均爲500 Lose 次數均爲0。在兩個負載圖中，藍白相間的線爲偵測系統一CPU 使用率的變化，而藍白相間的線爲偵測系統二CPU 使用率的變化。



我們可以得知，除了得知分流器有成功分流外，還可以了解到所有封包都集中的一台偵測系統上時CPU 的使用率大概是封包平均分流的二台偵測系統的二倍，因此分散式的NIDS 比單獨式的NIDS 更能適應於高速網路下的環境。

五、結果與討論

I. 結果

本計畫配合總計畫，進行 IP 位址追蹤技術的研究，我們不但對現有可行的技術進行研究，也實際在實驗環境中建置與實作了封包記錄系統與訊息採樣與 ICMP 追蹤系統，使我們對於 IP 追蹤的理論與實作上會發生的問題與可能的解決方式都有了更深入的了解，並且經由實際的程式寫作來了解系統底層的運作方式。

同時我們也針對實際系統在面對大量攻擊流量時所必需具備的分流技術進行研究，從而了解了各種方法的優缺點與其先天的限制，而且經由實驗獲得了保貴的實戰經驗。我們同時將計畫的成果，整理成三份教材，以供未來網管或網路安全人員參考使用。

II. 討論

我們對於進行這類整合行計畫的經驗不多，很感謝成大總計畫給予我們的指導與協助。本計畫的參與者利用一年的時間，在有限的經費下，做了最大的努力，對網路攻防上進行研究與開發。由於時間等種種因素，很多技術我們尚不能完全掌握，而且我們在測試的規模上也不夠擬真。對於這些技術，我們還必需要付出更出的努力才能達到整合並且自動化的最終目標。

參考文獻

[1] S. M. Bellovin. Security Problems in the TCP/IP Protocol Suite. ACM Computer Communications Review, 19(2):32–48, Apr. 1989.

[2] S. M. Bellovin. ICMP Traceback Messages. Internet Draft: draft-bellovin-itrace-00.txt, Oct. 2001.

[3] H. Burch and B. Cheswick. Tracing Anonymous Packets to Their Approximate Source. Unpublished paper, Dec. 1999.

[4] S. Deering. Internet protocol, version 6 (ipv6). RFC 2460, Dec. 1998.

[5] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing. RFC 2267, Jan. 1998.

[6] R. Govindan and H. Tangmunarunkit. Heuristics for Internet Map Discovery. In Proceedings of the 2000 IEEE INFOCOM Conference, Tel Aviv, Israel, Mar. 2000.

[7] C. Meadows. A Formal Framework and Evaluation Method for Network Denial of Service. In Proceedings of the 1999 IEEE Computer Security Foundations Workshop, Mordano,Italy, June 1999.

[8] O. Spatscheck and L. Peterson. Defending Against Denial of Service Attacks in Scout. In Proceedings of the 1999 USENIX/ACM Symposium on Operating System Design and Implementation, pages 59–72, Feb. 1999.

[9] R. Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. Proceedings of the 2000 USENIX Security Symposium, Denver, CO, July 2000.

[10] W. Theilmann and K. Rothermel. Dynamic Distance Maps of the Internet. In Proceedings of the 2000 IEEE INFOCOM Conference, Tel Aviv, Israel, Mar. 2000.

[11] Dave Dittrich. Distributed Denial of Service (DDoS) attacks/tools resource page. http://staff.washington.edu/dittrich/misc/ddos/.

[12] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. In Proceedings of the 2000 ACM SIGCOMM Conference, August 2000.

[13] K. Park and H. Lee. On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. In Proc. IEEE INFOCOM '01, pages 338-347, 2001.

[14] D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. Technical Report UCB/CSD-00-1107, University of California, Berkeley, June 2000.

[15] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. RFC 2474, Dec. 1995.

[16] D. Schnackenberg, K. Djahandari, and D. Sterne. Infrastructure for Intrusion Detection and Response. Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX), Hilton Head Island, SC, January 25-27, 2000.

[17] J. Postel. Internet Control Message Protocol. RFC 792, Internet Engineering Task Force, September 1981.

[18] John Ioannidis, Steven M. Bellovin. Proceedings of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, C alifornia 6-8 February 2002.

[19] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. “Hash-Based IP Traceback”. Proceedings of the ACM/SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '01), pp. 3-14, San Diego, CA, August 2001.

[20] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. “Single-Packet IP Traceback”. IEEE/ACM Transactions on Networking (ToN), Volume 10, Number 6, December 2002. Pages 721-734.

[21] SPIE User's Guide http://www.ir.bbn.com/projects/SPIE/UsersGuide.pdf

[22] Installing Freebsd, Mysql, and Snort Tutorial http://www.snort.org/docs/FreeBSD47RELEASE-Snort-MySQLVer1-3.pdf.

計畫成果自評

I. 原訂計畫目標：

本子計畫的目標在研發對應 DoS 及 DDoS 攻擊的追蹤技術，而參與此子計畫的台大與東華，則分別進行來源追蹤及分流攻擊偵測技術，針對這類攻擊大流量，攻擊點多的特性，進行技術的研究及系統的實作開發。

II. 研究內容與原計畫相符程度

完全符合。

III. 預期目標達成情況與綜合自評

我們對於分流偵測與來源追蹤技術進行研究與實作，本計畫除了讓參與計畫的研究生都有了良好的知識與實戰經驗，我們的成果還包括了最後的研發成果展示，以及資訊安全訓練教材。對於一個整合型計畫，能在不到一年的時間內達成這樣的成果，我們已盡了最大的努力，也謝謝成大的指導。

附錄 A: ICMP 訊息發送與流量統程式原始碼

（一）：產生 ICMP 追蹤封包的程式碼

- icmpd.h

```
#ifndef __ICMPD_H__

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define   MD5KEY         "The NBEN Project"

struct stamprecord {
     struct in_addr stamper;
     struct in_addr ip_src;
     struct in_addr ip_dst;
     unsigned long tv_sec;
     unsigned long tv_usec;
};

#define   UDP_DEFAULT_QUERY_PORT        36100
#define   UDP_NULL            0
#define   UDP_QUERY           1
#define   UDP_RESPONSE              2

#define   UDP_RESPONSE_NULL        0
#define   UDP_RESPONSE_REC_FOUND        1
#define   UDP_RESPONSE_REC_NOTFOUND     2

#define   UDP_OP_SIGNATURE         0x2be2dd05

struct udpop {
     unsigned int signature;
     unsigned int magic;
     unsigned short opcode;
     unsigned short result;
     struct in_addr ip_src;
     struct in_addr ip_dst;
     unsigned char md5sum[16];
};

#endif
```

- icmpd.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <signal.h>
#include <time.h>
#include <fcntl.h>
#include <pcap.h>
```

```
#include <sys/types.h>
/* md5 */
#ifdef LINUX
#include <openssl/md5.h>
#else
#include <md5.h>
#endif
/* ntohs */
#include <netinet/in.h>
/* inet_ntoa */
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
/* ether_ntoa */
#include <sys/socket.h>
#include <net/ethernet.h>
/* ip header */
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/udp.h>

#include "icmpd.h"

/* FreeBSD -> Linux */
#ifdef LINUX
#define   MD5Init        MD5_Init
#define MD5Update   MD5_Update
#define   MD5Final  MD5_Final
#else
char*
MD5(unsigned char *input, int len, unsigned char *output) {
     MD5_CTX   md5;
     MD5Init(&md5);
     MD5Update(&md5, (unsigned char *) input, len);
     MD5Final(output, &md5);
     return(output);
}
#endif

#define MAXPKTSIZE  1600

#define   ICMP_PKTSTAMP  32

pcap_t *p = NULL;
struct in_addr myid = {0};
FILE *logfp = NULL;
/* passing from parameters */
int noconsole = 0;
char *dev = NULL;
char *logfile = NULL;
int logsize = 10000;     /* in entries, default 10000 */
int logpos = 0;
int maxlog = 0;
int probability = 1000, magic = 7;
int qport = UDP_DEFAULT_QUERY_PORT;
int qsocket = -1;
```

```
extern int errno;

void cbreak(int s) {
     fprintf(stderr, "** break\n");
     if(logfp) {
          fclose(logfp);
     }
     if(p) {
          pcap_close(p);
     }
}

#if 1     // SEND ICMP
struct icmp {
     u_char    icmp_type;
     u_char    icmp_code;
     u_short   icmp_sum;
     /* private message */
     u_int     unused;
     u_int     stamper;
     u_int     srcip;
     u_int     dstip;
     long tv_sec;
     long tv_usec;
     u_char    md5[16];
};

int
init_raw() {
     int fd, optval = 1;

     if((fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW))==-1)
          return(-1);

     if(setsockopt(fd, IPPROTO_IP, IP_HDRINCL, &optval, sizeof(optval))==-1) {
          close(fd);
          return(-1);
     }

     return(fd);
}

void
send_raw(struct in_addr ip_dst, u_char *raw, int len) {
     static int fd = -1;
     struct sockaddr_in sin;

     if(fd < 0) {
          if((fd = init_raw())==-1) {
               fprintf(stderr, "raw: cannot init raw socket.\n");
               return;
          }
     }

     bzero(&sin, sizeof(sin));
     sin.sin_family = AF_INET;
     sin.sin_addr.s_addr = ip_dst.s_addr;
```

```
     if(sendto(fd, raw, len, 0, (struct sockaddr*) &sin, sizeof(sin)) < 0) {
          fprintf(stderr, "raw: sendto() failed(%d): %s.\n", errno, strerror(errno));
          return;
     }

#if 0
     fprintf(stderr, "raw: send icmp to %s\n", inet_ntoa(ip_dst));
#endif
}

unsigned short
in_cksum(unsigned short *addr, int len) {
     register int sum = 0;
     u_short answer = 0;
     register u_short *w = addr;
     register int nleft = len;

     while (nleft > 1)  {
          sum += *w++;
          nleft -= 2;
     }

     /* mop up an odd byte, if necessary */
     if (nleft == 1) {
          *(u_char *)(&answer) = *(u_char *)w ;
          sum += answer;
     }

     /* add back carry outs from top 16 bits to low 16 bits */
     sum = (sum >> 16) + (sum & 0xffff);     /* add hi 16 to low 16 */
     sum += (sum >> 16);                     /* add carry */
     answer = ~sum;                          /* truncate to 16 bits */
     return(answer);
}

void
send_icmp(long tv_sec, long tv_usec, struct in_addr ip_src, struct in_addr ip_dst) {
     u_char buf[MAXPKTSIZE];
     struct ip *ip;
     struct icmp *icmp;
     int pktsize;

     pktsize = sizeof(struct ip) + sizeof(struct icmp);
     bzero(buf, pktsize);
     ip = (struct ip*) buf;
     icmp = (struct icmp*) (buf + sizeof(struct ip));

     ip->ip_hl = 5;           /* 5*4 == 20 bytes */
     ip->ip_v  = 4;
     ip->ip_tos     = 0;
#ifdef LINUX
     ip->ip_len     = htons(pktsize);
#else
     ip->ip_len     = pktsize;          /* in FreeBSD: do NOT use htons() */
#endif
     ip->ip_id = rand() % 0x0ffff;
```

```
     ip->ip_off     = 0;
     ip->ip_ttl     = 0x7f;
     ip->ip_p  = IPPROTO_ICMP;          /* ICMP */
     ip->ip_sum     = 0;
     ip->ip_src     = myid;
     ip->ip_dst     = ip_dst;
     ip->ip_sum     = in_cksum((unsigned short *) ip, sizeof(struct ip));

     icmp->icmp_type     = ICMP_PKTSTAMP;
     icmp->icmp_code     = 0;
     icmp->icmp_sum = 0;
     icmp->unused   = 0;
     icmp->stamper  = myid.s_addr;
     icmp->srcip    = ip_src.s_addr;
     icmp->dstip    = ip_dst.s_addr;
     icmp->tv_sec   = tv_sec;
     icmp->tv_usec  = tv_usec;
     /* md5 */
     do {
          MD5_CTX   md5;
          MD5Init(&md5);
          MD5Update(&md5, (unsigned char *) icmp, sizeof(struct icmp) - 16);
          MD5Update(&md5, (unsigned char*) MD5KEY, strlen(MD5KEY));
          MD5Final(icmp->md5, &md5);
     } while(0);
     icmp->icmp_sum = in_cksum((unsigned short *) icmp, sizeof(struct icmp));

     send_raw(ip_dst, buf, pktsize);
}
#endif    // SEND ICMP

unsigned char *vector = NULL;

void
bloom_filter(unsigned int ip_src, unsigned int ip_dst)
{
     //printf("enter\n");
     int i=0;
     static int first = 0;
     u_int src_dst_address[2];
     u_int output[4];// sizeof(u_int) = 4

     if (first == 0) {
          first = 1;
          // allocate 28 bit vector
          vector = malloc(1024*1024*256);
          if (vector == NULL) {
               printf("no enough memory space, so no bloom filter\n");
               first = 0;
               return;
          }
     }

     src_dst_address[0] = ip_src;
     src_dst_address[1] = ip_dst;
     MD5((unsigned char *)src_dst_address, 2*sizeof(u_int), (unsigned char *)output);
```

```
     for (i=0; i<4; i++) {
          unsigned char *position = vector;
          unsigned char mask = 1;

          output[i] &= 0x0fffffff;
          position += ((output[i])>>3);
          mask <<= (output[i] & 0x07);
          (*position) |= mask;
     }
}

int
query_bloom(u_int src, u_int dst) {
     int i;
     u_int src_dst_address[2];
     u_int output[4];// sizeof(u_int) = 4

     if(vector) {
          src_dst_address[0] = src;
          src_dst_address[1] = dst;
          MD5((unsigned char *)src_dst_address, 2*sizeof(u_int), (unsigned char *)output);

          for (i=0; i<4; i++) {
               unsigned char *position = vector;
               unsigned char mask = 1;

               output[i] &= 0x0fffffff;
               position += ((output[i])>>3);
               mask <<= (output[i] & 0x07);
               if(((*position) & mask)==0)
                    break;
          }

          if(i >= 4)
               return(1);
     }

     return(0);
}

unsigned short
process_ethernet(u_char *user, const struct pcap_pkthdr *hdr, const u_char *payload) {
     u_int caplen = hdr->caplen;
     //u_int length = hdr->len;
     struct ether_header *eptr;  /* net/ethernet.h */
     u_short ether_type;

     if (caplen < ETHER_HDR_LEN) {
          /* not a ethernet packet */
          if(noconsole==0)
               fprintf(stdout, "* not an etherned packet, too short\n");
          return -1;
     }

     /* lets start with the ether header... */
     eptr = (struct ether_header *) payload;
     ether_type = ntohs(eptr->ether_type);
```

```
     return(ether_type);
}

int process_ip(u_char *user, const struct pcap_pkthdr *hdr, const u_char *payload) {
     const struct ip *ip;

     /* jump pass the ethernet header */
     ip = (struct ip*) (payload + sizeof(struct ether_header));

     if (hdr->len < sizeof(struct ip)) {     /* bad length, packet too short */
          //fprintf(stdout, "* not an ip packet, too short\n");
          return(-1);
     }

     if(ip->ip_v != 4) {           /* bad version */
          //fprintf(stdout, "* bad ip version\n");
          return(-1);
     }

     if(ip->ip_hl < 5) {                /* bad header length */
          //fprintf(stdout, "* not an ip packet, header too short\n");
          return(-1);
     }

     if(ip->ip_p==IPPROTO_UDP) {
          struct udphdr *udp;
          struct udpop *opdata, oprep;
          struct sockaddr_in sin;

          udp = (struct udphdr*) (payload + sizeof(struct ether_header) + ((ip->ip_hl) <<
2));
          opdata = (struct udpop*) (payload + sizeof(struct ether_header) + ((ip->ip_hl)
<< 2) + sizeof(*udp));

#if 0
          if(ntohs(udp->uh_dport) > 2000)
          fprintf(stderr, "got UDP from %s: dport=%d len=%d signature=0x%08x!\n",
inet_ntoa(ip->ip_src),
                    ntohs(udp->uh_dport), ntohs(udp->uh_ulen),
ntohl(opdata->signature));
#endif

          if(ip->ip_dst.s_addr==myid.s_addr)
          if(ntohs(udp->uh_dport) == qport) {
          /********************/
               int slen = sizeof(sin);
               recvfrom(qsocket, &oprep, sizeof(oprep), 0, (struct sockaddr*) &sin,
&slen);
          if(ntohs(udp->uh_ulen) == sizeof(struct udpop)+sizeof(struct udphdr))
          if(ntohl(opdata->signature) == UDP_OP_SIGNATURE) {
               unsigned char md5str[16];
               MD5_CTX   md5;

               MD5Init(&md5);
               MD5Update(&md5, (unsigned char *) opdata, sizeof(struct udpop) - 16);
               MD5Update(&md5, (unsigned char*) MD5KEY, strlen(MD5KEY));
```

```
                MD5Final(md5str, &md5);

                if(memcmp(md5str, opdata->md5sum, 16)==0) {
                     if(ntohs(opdata->opcode)==UDP_QUERY) {
                          int ret;

                          bcopy(opdata, &oprep, sizeof(oprep)-16);
                          oprep.opcode = htons(UDP_RESPONSE);
                          if(query_bloom(opdata->ip_src.s_addr, opdata->ip_dst.s_addr) >
0) {
                               oprep.result = htons(UDP_RESPONSE_REC_FOUND);
                          } else {
                               oprep.result = htons(UDP_RESPONSE_REC_NOTFOUND);
                          }
                          MD5Init(&md5);
                          MD5Update(&md5, (unsigned char *) &oprep, sizeof(oprep) - 16);
                          MD5Update(&md5, (unsigned char*) MD5KEY, strlen(MD5KEY));
                          MD5Final(md5str, &md5);
                          bcopy(md5str, oprep.md5sum, 16);

                          bzero(&sin, sizeof(sin));
                          sin.sin_family = AF_INET;
                          sin.sin_addr = ip->ip_src;
                          sin.sin_port = udp->uh_sport;
#ifndef LINUX
                          sin.sin_len = sizeof(sin);
#endif
                          sendto(qsocket, &oprep, sizeof(oprep), 0, (struct sockaddr*)
&sin, sizeof(sin));

                          ret = htons(oprep.result);
                          fprintf(stderr, "query 0x%08x->0x%08x from %s, ret=%d (%s).\n",
                                    ntohl(opdata->ip_src.s_addr),
ntohl(opdata->ip_dst.s_addr),
                                    inet_ntoa(ip->ip_src), ret,
                                    ret==1 ? "found" : (ret==2 ? "not found" : "unknown"));
                     }
                }
          }
          /********************/
          }
     } else if(ip->ip_p==IPPROTO_ICMP) {
          struct icmp *icmp;
          icmp = (struct icmp*) (payload + sizeof(struct ether_header) + ((ip->ip_hl) <<
2));
          if(in_cksum((unsigned short *) icmp, sizeof(struct icmp))==0)
          if(icmp->icmp_type==ICMP_PKTSTAMP) {
               char id[24], srcip[24];
               char md5str[16], md5strx[16];
               MD5_CTX   md5;

               bcopy(icmp->md5, md5str, 16);
               bzero(icmp->md5, 16);
               icmp->icmp_sum = 0;

               MD5Init(&md5);
               MD5Update(&md5, (unsigned char *) icmp, sizeof(struct icmp) - 16);
```

```
               MD5Update(&md5, (unsigned char*) MD5KEY, strlen(MD5KEY));
               MD5Final(md5strx, &md5);

               if(memcmp(md5str, md5strx, 16)==0) {
                    struct stamprecord rec;

                    bzero(&rec, sizeof(rec));
                    rec.stamper.s_addr = icmp->stamper;
                    rec.ip_src.s_addr  = icmp->srcip;
                    rec.ip_dst.s_addr  = icmp->dstip;
                    rec.tv_sec        = icmp->tv_sec;
                    rec.tv_usec       = icmp->tv_usec;

                    if(icmp->stamper==myid.s_addr)
                         return(0);

                    if(logfp) {
                         fwrite(&rec, sizeof(rec), 1, logfp);
                         fflush(logfp);
                         logpos = ftell(logfp);
                         if(logpos >= maxlog) {
                              fseek(logfp, 0, SEEK_SET);
                         }
                    }

                    snprintf(id, 23, inet_ntoa(rec.stamper));
                    snprintf(srcip, 23, inet_ntoa(rec.ip_src));

                    if(noconsole==0)
                         fprintf(stdout, "STAMP: id=%s, src=%s, dst=%s,
t=%10ld.%-6ld.\n",
                              id, srcip, inet_ntoa(rec.ip_dst), rec.tv_sec,
rec.tv_usec);

                    return(0);
               }

               if(noconsole==0)
                    fprintf(stdout, "STAMP: bad md5sum!\n");
          }
     }

     do {                    /* output */
#if 0
          char srcip[24];
          snprintf(srcip, 23, inet_ntoa(ip->ip_src));
          fprintf(stdout, "%10ld.%-6ld %15s -> %-15s\n",
               hdr->ts.tv_sec, hdr->ts.tv_usec,
               srcip, inet_ntoa(ip->ip_dst));
#endif
          /* add bloom filter here*/
          bloom_filter(ip->ip_src.s_addr, ip->ip_dst.s_addr);

          if((rand()%probability)==magic)
               send_icmp(hdr->ts.tv_sec, hdr->ts.tv_usec, ip->ip_src, ip->ip_dst);
     } while(0);
```

```
     return(0);
}

void
handler(u_char *user, const struct pcap_pkthdr *hdr, const u_char *payload) {
     unsigned short type = process_ethernet(user, hdr, payload);

     if(type == ETHERTYPE_IP) {
          process_ip(user, hdr, payload);
     }
}

void usage(char *progname) {
     fprintf(stderr, "usage: %s -i dev -m myid [-l logfile] [-s logsize] [-p prob] [-q port]
[-x]\n"
               "\t-i: specify binding device\n"
               "\t-m: specify router identifier\n"
               "\t-l: specify log filename\n"
               "\t-s: specify log filesize (in entries), default = 10000.\n"
               "\t-p: specify generating probability, default = 1/1000.\n"
               "\t-q: specify udp query port, default = 36100.\n"
               "\t-x: no console output\n",
               progname);
}

int
main(int argc, char *argv[]) {
     int ch;
     char errbuf[PCAP_ERRBUF_SIZE];
     char *progname = argv[0];

     srand(time(0));

     while((ch = getopt(argc, argv, "i:m:s:l:p:q:x"))!=-1) {
          switch(ch) {
          case 'i':
               dev = strdup(optarg);
               break;
          case 'm':
               myid.s_addr = inet_addr(optarg);
               break;
          case 'l':
               logfile = strdup(optarg);
               break;
          case 's':
               logsize = atoi(optarg);
               break;
          case 'p':
               probability = atoi(optarg);
               magic = rand() % probability;
               break;
          case 'q':
               qport = atoi(optarg);
               break;
          case 'x':
               noconsole = 1;
               break;
```

```
          case '?':
          default:
               usage(progname);
               return(-1);
          }
     }
     argc -= optind;
     argv += optind;

     if(dev==NULL) {
          fprintf(stderr, "error: no device specified.\n");
          usage(progname);
          return(-1);
     }

     if(myid.s_addr==0) {
          fprintf(stderr, "error: no id specified.\n");
          usage(progname);
          return(-1);
     }

     if((p = pcap_open_live(dev, MAXPKTSIZE, 1, 1000, errbuf))==NULL) {
          fprintf(stderr, "pcap open failed: %s\n", errbuf);
          return(-1);
     }

     if(qport > 0) {
          int fl;
          struct sockaddr_in sin;

          if((qsocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))==-1) {
               fprintf(stderr, "socket() failed: %s\n", strerror(errno));
               return(-1);
          }

          bzero(&sin, sizeof(sin));
          sin.sin_family = AF_INET;
          //sin.sin_addr = myid;
          sin.sin_port = htons(qport);
          if(bind(qsocket, (struct sockaddr*) &sin, sizeof(sin))==-1) {
               fprintf(stderr, "bind() on port %d failed: %s\n", qport, strerror(errno));
               return(-1);
          }

          if((fl = fcntl(qsocket, F_GETFL, NULL))==-1)
               fl = 0;
          fcntl(qsocket, F_SETFL, fl|O_NONBLOCK);
     }

     if(logfile) {
          struct stamprecord rec;

          maxlog = logsize * sizeof(rec);

          if((logfp = fopen(logfile, "r+"))==NULL) {
               if((logfp = fopen(logfile, "w"))==NULL) {
                    fprintf(stderr, "cannot open logfile\n");
```

```
                    goto exit_icmpd;
               }
          }

          fprintf(stderr, "logmax: %d\n", maxlog);

          fseek(logfp, 0, SEEK_END);
          logpos = ftell(logfp);

          if(logpos >= maxlog) {
               int len;
               long last = 0;
               fseek(logfp, 0, SEEK_SET);
               while((len = fread(&rec, sizeof(rec), 1, logfp)) > 0) {
                    if(rec.tv_sec <= last)
                         break;
                    last = rec.tv_sec;
               }
               logpos = ftell(logfp);
               if(logpos > 0 && logpos < maxlog)
                    logpos -= sizeof(rec);
               else
                    logpos = 0;
               fseek(logfp, 0, SEEK_SET);
          }

          fprintf(stderr, "log: starting from %d\n", logpos/sizeof(rec));
     }

#if 0
     if(errbuf[0]!='\0')
          fprintf(stderr, "warning: %s\n", errbuf);
#endif
     fprintf(stderr, "dev=%s id[%s] p=1/%d", dev, inet_ntoa(myid), probability);
     if(logfp)
          fprintf(stderr, " logfile=%s (max=%d)", logfile, logsize);
     fprintf(stderr, "%s.\n", noconsole ? ", console disabled" : "");
     if(qport > 0)
          fprintf(stderr, "enable query on udp port %d\n", qport);

     signal(SIGTERM, cbreak);

     pcap_loop(p, -1, handler, NULL);

     if(logfp)
          fclose(logfp);

exit_icmpd:
     pcap_close(p);

     return(0);
}
```

（二）：查詢封包真正來源的程式碼

- iquery.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <fcntl.h>
#include <sys/types.h>
/* md5 */
#ifdef LINUX
#include <openssl/md5.h>
#else
#include <md5.h>
#endif
/* ntohs */
#include <netinet/in.h>
/* inet_ntoa */
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#include "icmpd.h"

/* FreeBSD -> Linux */
#ifdef LINUX
#define   MD5Init        MD5_Init
#define MD5Update   MD5_Update
#define   MD5Final  MD5_Final
#else
char*
MD5(unsigned char *input, int len, unsigned char *output) {
     MD5_CTX   md5;
     MD5Init(&md5);
     MD5Update(&md5, (unsigned char *) input, len);
     MD5Final(output, &md5);
     return(output);
}
#endif

extern int errno;

#define MAX_STAMPER 32767
#define   MAX_TRIES 3

struct stamper {
     struct in_addr id;
     int count;
};

int totals = 0;
struct stamper all[MAX_STAMPER];

int
id_comp(const void *a, const void *b) {
     struct stamper *sa, *sb;
```

```
     sa = (struct stamper*) a;
     sb = (struct stamper*) b;
     if(sa->id.s_addr == sb->id.s_addr)
          return(0);
     if(sa->id.s_addr > sb->id.s_addr)
          return(1);
     return(-1);
}

int
c_comp(const void *a, const void *b) {
     struct stamper *sa, *sb;
     sa = (struct stamper*) a;
     sb = (struct stamper*) b;
     if(sa->count== sb->count)
          return(0);
     if(sa->count < sb->count)
          return(1);
     return(-1);
}

void
add_stamper(unsigned int id) {
     struct stamper *s, tmp;
     tmp.id.s_addr = id;
     tmp.count = -1;
     if((s = bsearch(&tmp, all, totals, sizeof(struct stamper), id_comp))!=NULL) {
          s->count++;
     } else {
          if(totals < MAX_STAMPER) {
               all[totals].id.s_addr = id;
               all[totals].count = 1;
               totals++;
               qsort(all, totals, sizeof(struct stamper), id_comp);
          } else {
               fprintf(stderr, "too much stampers.\n");
          }
     }
}

void
dump_stamper(int count) {
     int i;

     qsort(all, totals, sizeof(struct stamper), c_comp);

     fprintf(stdout, "+-------+----------+-----------------+\n"
               "|   ## |   count | stamper        |\n"
               "+-------+----------+-----------------+\n");
     for(i = 0; i < totals; i++)
          fprintf(stdout, "| %5d | %8d | %-15s |\n", i+1, all[i].count,
inet_ntoa(all[i].id));
     fprintf(stdout, "+-------+----------+-----------------+\n");
     fprintf(stdout,     "+         %8d entries parsed. |\n", count);
     fprintf(stdout, "+-----------------------------------+\n");
}
```

```
int qsocket = -1;

int
do_query(unsigned int ip_src, unsigned int ip_dst, unsigned int target) {
     int i;
     struct udpop query, response;
     struct sockaddr_in sin, rsin;

     if(qsocket==-1) {
          struct timeval optval;

          if((qsocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))==-1) {
               fprintf(stderr, "socket() failed: %s\n", strerror(errno));
               exit(-1);
          }

          optval.tv_sec = 5;
          optval.tv_usec = 0;
          setsockopt(qsocket, SOL_SOCKET, SO_RCVTIMEO, &optval, sizeof(optval));
     }

     query.signature = htonl(UDP_OP_SIGNATURE);
     query.magic = rand() & 0xffffffff;
     query.opcode = htons(UDP_QUERY);
     query.result = htons(UDP_RESPONSE_NULL);
     query.ip_src.s_addr = ip_src;
     query.ip_dst.s_addr = ip_dst;
     do { /* MD5 */
          MD5_CTX md5;
          MD5Init(&md5);
          MD5Update(&md5, (unsigned char *) &query, sizeof(query)-16);
          MD5Update(&md5, (unsigned char*) MD5KEY, strlen(MD5KEY));
          MD5Final(query.md5sum, &md5);
     } while(0);

     bzero(&sin, sizeof(sin));
     sin.sin_family = AF_INET;
     sin.sin_addr.s_addr = target;
     sin.sin_port = htons(UDP_DEFAULT_QUERY_PORT);
#ifndef LINUX
     sin.sin_len = sizeof(sin);
#endif
     for(i = 0; i < MAX_TRIES; i++) {
          int slen = sizeof(rsin);
          if(sendto(qsocket, &query, sizeof(query), 0, (struct sockaddr*) &sin,
sizeof(sin))==-1) {
               fprintf(stdout, " txx");
               continue;
          }
          bzero(&rsin, sizeof(rsin));
          if(recvfrom(qsocket, &response, sizeof(response), 0, (struct sockaddr*) &rsin,
&slen)==-1) {
               fprintf(stdout, " rxx");
               continue;
          }
          if(response.signature==htonl(UDP_OP_SIGNATURE)
          && response.opcode==ntohs(UDP_RESPONSE)
```

```
          && response.magic==query.magic) {
               return(ntohs(response.result));
          }
     }
     return(0);
}

int main(int argc, char *argv[]) {
     int i, ret, count = 0;
     unsigned int ip_src, ip_dst;
     char *p, *target = NULL;
     struct stamprecord rec;
     FILE *fp;

     if(argc < 3) {
          fprintf(stderr, "usage: %s query-pair logfile|target\n", argv[0]);
          return(-1);
     }

     ip_src = ip_dst = 0;
     if((p = strtok(argv[1], ":"))!=NULL) {
          ip_src = inet_addr(p);
          if((p = strtok(NULL, ":"))!=NULL) {
               ip_dst = inet_addr(p);
          } else {
               fprintf(stderr, "cannot get q-dst.\n");
               return(-1);
          }
     } else {
          fprintf(stderr, "cannot get q-src.\n");
          return(-1);
     }

     if((fp = fopen(argv[2], "r"))==NULL) {
          target = strdup(argv[2]);
          if(inet_addr(target)==INADDR_NONE) {
               fprintf(stderr, "'%s' is neigher a file nor an invalid address.\n", argv[2]);
               return(-1);
          }
     }

     if(target==NULL) {
     /*********************************/
     // read log
     while(fread(&rec, sizeof(rec), 1, fp) > 0) {
          add_stamper(rec.stamper.s_addr);
          count++;
     }

     fclose(fp);

     dump_stamper(count);

     srand(time(0));
     for(i = 0; i < totals; i++) {
          fprintf(stdout, "query %s", inet_ntoa(*((struct in_addr *) &ip_src)));
          fprintf(stdout, "->%s from ", inet_ntoa(*((struct in_addr *) &ip_dst)));
```

```
          fprintf(stdout, "%s ...", inet_ntoa(all[i].id));
          if((ret = do_query(ip_src, ip_dst, all[i].id.s_addr)) > 0) {
               if(ret==UDP_RESPONSE_REC_FOUND)
                    fprintf(stdout, " [FOUND]\n");
               else if(ret==UDP_RESPONSE_REC_NOTFOUND)
                    fprintf(stdout, " [NOT FOUND]\n");
               else
                    fprintf(stdout, " [UNKNOWN]\n");
          } else {
               fprintf(stdout, " [FAILED]\n");
          }
     }
     /*********************************/
     } else {
          fprintf(stdout, "query 0x%08x->0x%08x from %s ...", ntohl(ip_src),
ntohl(ip_dst),
                    target);
          if((ret = do_query(ip_src, ip_dst, inet_addr(target))) > 0) {
               if(ret==UDP_RESPONSE_REC_FOUND)
                    fprintf(stdout, " [FOUND]\n");
               else if(ret==UDP_RESPONSE_REC_NOTFOUND)
                    fprintf(stdout, " [NOT FOUND]\n");
               else
                    fprintf(stdout, " [UNKNOWN]\n");
          } else {
               fprintf(stdout, " [FAILED]\n");
          }
     }

     fprintf(stdout, "done.\n");

     return(0);
}
```

附錄 B: 分流系統程式原始碼

Packet generator

```
#include <netinet/ip.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/tcp.h>
#include <getopt.h>
#include <sys/types.h>
#include <netdb.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <math.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
inline long getrandom (int, int);
void random_init (void);
unsigned short ip_sum (unsigned short *, int);
unsigned short cksum (unsigned short *, int);
unsigned short checksum (unsigned short *, int);
int isip (char *);
unsigned long resolve (char *);
char rseed[65535];
int rcounter;
int main (int argc, char **argv)
{
unsigned short seqrandom = 1;
unsigned short ackrandom = 1;
unsigned short idrandom =1 ;
unsigned short sportrandom = 1;
unsigned short dportrandom = 1;
u_int16_t urgflag=0x0;
u_int16_t ackflag=0x0;
u_int16_t pshflag=0x0;
```

```
u_int16_t rstflag=0x0;
u_int16_t synflag=0x0;
u_int16_t finflag=0x0;
unsigned long count = 1;
unsigned long loop = 0;
unsigned long interval = 200; //200ns
u_int8_t ttl = 0x20;
u_int16_t id = 0x1;
u_int16_t frag = 0x4000; // DF
u_int8_t tos = 0x0;
unsigned long ipsrc = -33511232; // 192.168.0.254
unsigned long ipdst = 184789184; // 192.168.3.11
u_int16_t sport = 1234;
u_int16_t dport = 80;
u_int32_t seq = 0x1;
u_int32_t ack = 0x1020;
int c;
int digit_optind = 0;
unsigned long chsum = 0;
struct sockaddr_in pothead;
struct iphdr *iph;
struct tcphdr *tcph;
int sockfd;
char *packet;
int pktsize = sizeof (struct iphdr) + sizeof (struct tcphdr);
while (1)
{
int this_option_optind = optind ? optind : 1;
int option_index = 0;
static struct option long_options[] = {
{"help", 0, 0, 0},
{"dport", 1, 0, 0},
{"sport", 1, 0, 0},
{"seq", 1, 0, 0},
{"ack", 1, 0, 0},
{"syn", 0, 0, 0},
{"urg", 0, 0, 0},
{"rst", 0, 0, 0},
{"psh", 0, 0, 0},
```

```
{"fin", 0, 0, 0},
{0, 0, 0, 0}
};
c = getopt_long (argc, argv, "c:d:f:i:s:p:q:t:v:", long_options, &option_index);
if (c == -1)
break;
switch (c)
{
case 0:
if (strcmp (long_options[option_index].name, "help") == 0)
{
fprintf(stderr,"./transp [--option1] [-option2]\n");
fprintf(stderr,"option1:\n");
fprintf(stderr,"--sport arg : specify the source port\n");
fprintf(stderr,"--dport arg : specify the destination port\n");
fprintf(stderr,"--seq arg : specify the sequence field in TCP header\n");
fprintf(stderr,"--ack arg : specify the acknowledge field in TCP header and enable the ack
flag\n");
fprintf(stderr,"--urg : enable the urg flag in TCP header\n");
fprintf(stderr,"--psh : enable the psh flag in TCP header\n");
fprintf(stderr,"--rst : enable the rst flag in TCP header\n");
fprintf(stderr,"--syn : enable the syn flag in TCP header\n");
fprintf(stderr,"--fin : enable the fin flag in TCP header\n");
fprintf(stderr,"--help arg : you will see this\n");
fprintf(stderr,"option2:\n");
fprintf(stderr,"-c arg : how many packet do you want to transit\n");
fprintf(stderr,"-d arg : specify the destionation IP field in IP header\n");
fprintf(stderr,"-i arg : specify the identificatioin field in IP header\n");
fprintf(stderr,"-s arg : specify the source IP field in IP header\n");
fprintf(stderr,"-p arg : specify the payload length after IP header\n");
fprintf(stderr,"-q [T|R|C] : specify the tos field in IP header\n");
fprintf(stderr," T : stands for max throughput\n");
fprintf(stderr," R : stands for max relibility\n");
fprintf(stderr," C : stands for min cost\n");
fprintf(stderr,"-t arg : specify the ttl field in IP headre\n");
fprintf(stderr,"-v arg : specify the interval of each packet\n");
}
if (strcmp (long_options[option_index].name, "sport") == 0)
{ sport = atoi (optarg);
```

```
sportrandom = 0;
}
if (strcmp (long_options[option_index].name, "dport") == 0)
{ dport = atoi (optarg);
dportrandom =0;
}
if (strcmp (long_options[option_index].name, "seq") == 0)
{
seq=atoi (optarg);
seqrandom=0;
}
if (strcmp (long_options[option_index].name, "ack") == 0)
{
ack=atoi(optarg);
ackflag=0x1;
ackrandom=0;
}
if (strcmp (long_options[option_index].name, "syn") == 0)
{
synflag=1;
}
if (strcmp (long_options[option_index].name, "rst") == 0)
{
rstflag=1;
}
if (strcmp (long_options[option_index].name, "psh") == 0)
{
pshflag=1;
}
if (strcmp (long_options[option_index].name, "urg") == 0)
{
urgflag=1;
}
if (strcmp (long_options[option_index].name, "fin") == 0)
{
finflag=1;
}
break;
case 'c':
```

```
count = atoi (optarg);
break;
case 'd':
ipdst = resolve (optarg);
break;
case 'f':
if (strncmp (optarg, "R", 1) == 0)
frag = 0x8000;
else if (strncmp (optarg, "D", 1) == 0)
frag = 0x4000;
else if (strncmp (optarg, "M", 1) == 0)
frag = 0x2000;
break;
case 'i':
id = atoi (optarg);
idrandom = 0;
break;
case 's':
ipsrc = resolve (optarg);
break;
case 'p':
pktsize+=atoi(optarg);
break;
case 'q':
if (strncmp (optarg, "D", 1) == 0)
tos = 0x10;
else if (strncmp (optarg, "T", 1) == 0)
tos = 0x08;
else if (strncmp (optarg, "R", 1) == 0)
tos = 0x04;
else if (strncmp (optarg, "C", 1) == 0)
tos = 0x02;
break;
case 't':
ttl = atoi (optarg);
break;
case 'v':
interval = atoi (optarg);
break;
```

```
}
}
packet = (char *) malloc (sizeof (char) * pktsize);
sockfd = socket (AF_INET, SOCK_RAW, 255);
iph = (struct iphdr *) (packet);
tcph = (struct tcphdr *) (packet + sizeof (struct iphdr));
memset (packet, 0, pktsize);
iph->version = 4;
iph->ihl = 5;
iph->tos = tos;
iph->tot_len = htons (pktsize);
iph->id = htons (id);
iph->frag_off = htons (frag);
iph->ttl = ttl;
iph->protocol = 0x6;
// iph->saddr = inet_addr ("192.168.0.254");
iph->saddr = ipsrc;
// iph->daddr = inet_addr ("192.168.3.11");
iph->daddr = ipdst;
iph->check = ip_sum ((short int *) &iph, sizeof (iph));
tcph->source = htons (sport);
tcph->dest = htons (dport);
tcph->seq = htonl (seq);
tcph->ack_seq = htonl (ack);
tcph->doff = 5;
tcph->urg=urgflag;
tcph->ack=ackflag;
tcph->psh=pshflag;
tcph->rst=rstflag;
tcph->syn=synflag;
tcph->fin=finflag;
tcph->window = htons (1024);
chsum += htons (tcph->source);
chsum += htons (tcph->dest);
chsum += (short) (htonl (tcph->seq) >> 16);
chsum += (short) (htonl (tcph->seq) & 0x0000ffff);
chsum += (short) (htonl (tcph->ack_seq) >> 16);
chsum += (short) (htonl (tcph->ack_seq) & 0x0000ffff);
chsum += 0x5002;
```

```
chsum += htons (tcph->window);
chsum += (short) (htonl (iph->saddr) >> 16);
chsum += (short) (htonl (iph->saddr) & 0x0000ffff);
chsum += (short) (htonl (iph->daddr) >> 16);
chsum += (short) (htonl (iph->daddr) & 0x0000ffff);
chsum += iph->protocol;
chsum += 20;
chsum = (chsum >> 16) + (chsum & 0xffff);
(unsigned short) chsum += (chsum >> 16);
tcph->check = ~htons (chsum) - 512;
pothead.sin_family = AF_INET;
pothead.sin_addr.s_addr = iph->daddr;
setsockopt (sockfd, IPPROTO_IP, IP_HDRINCL, "1", sizeof ("1"));
for (loop = 0; loop < count; loop++)
{
if (seqrandom == 1)
tcph->seq = htonl (getrandom(1,65535));
if (ackrandom == 1)
tcph->ack_seq = htonl (getrandom(1,65535));
if (idrandom == 1)
iph->id = htons (getrandom(1,65535));
if (sportrandom==1)
tcph->source = htons (getrandom(1024,65535));
if (dportrandom ==1)
{
dport = getrandom(1,65535);
tcph->dest = htons (dport);
pothead.sin_port = htons (dport);
}
chsum=0;
chsum += htons (tcph->source);
chsum += htons (tcph->dest);
chsum += (short) (htonl (tcph->seq) >> 16);
chsum += (short) (htonl (tcph->seq) & 0x0000ffff);
chsum += (short) (htonl (tcph->ack_seq) >> 16);
chsum += (short) (htonl (tcph->ack_seq) & 0x0000ffff);
chsum += 0x5002;
chsum += htons (tcph->window);
chsum += (short) (htonl (iph->saddr) >> 16);
```

```
chsum += (short) (htonl (iph->saddr) & 0x0000ffff);
chsum += (short) (htonl (iph->daddr) >> 16);
chsum += (short) (htonl (iph->daddr) & 0x0000ffff);
chsum += iph->protocol;
chsum += 20;
chsum = (chsum >> 16) + (chsum & 0xffff);
(unsigned short) chsum += (chsum >> 16);
tcph->check = ~htons (chsum) - 512;
usleep (interval);
sendto (sockfd, packet, pktsize, 0, (struct sockaddr_in *)
&pothead,
sizeof (struct sockaddr_in));
}
free (packet);
exit(0);
}
unsigned short
checksum (addr, len)
unsigned short *addr;
int len;
{
register int nleft = len;
register unsigned short *w = addr;
register int sum = 0;
unsigned short answer = 0;
while (nleft > 1)
{
sum += *w++;
nleft -= 2;
}
if (nleft == 1)
{
*(unsigned char *) (&answer) = *(unsigned char *) w;
sum += answer;
}
return (sum);
}
unsigned short
ip_sum (addr, len)
```

```
unsigned short *addr;
int len;
{
register int nleft = len;
register unsigned short *w = addr;
register int sum = 0;
unsigned short answer = 0;
while (nleft > 1)
{
sum += *w++;
nleft -= 2;
}
if (nleft == 1)
{
*(unsigned char *) (&answer) = *(unsigned char *) w;
sum += answer;
}
sum = (sum >> 16) + (sum & 0xffff);
sum += (sum >> 16);
answer = ~sum;
return (answer);
}
unsigned short
cksum (unsigned short *buf, int nwords)
{
unsigned long sum;
for (sum = 0; nwords > 0; nwords--)
sum += *buf++;
sum = (sum >> 16) + (sum & 0xffff);
sum += (sum >> 16);
return ~sum;
}
int
isip (char *ip)
{
int a, b, c, d;
sscanf (ip, "%d.%d.%d.%d", &a, &b, &c, &d);
if (a < 0)
return 0;
```

```
if (a > 255)
return 0;
if (b < 0)
return 0;
if (b > 255)
return 0;
if (c < 0)
return 0;
if (c > 255)
return 0;
if (d < 0)
return 0;
if (d > 255)
return 0;
return 1;
}
unsigned long
resolve (char *host)
{
struct hostent *he;
struct sockaddr_in tmp;
if (isip (host))
return (inet_addr (host));
he = (struct hostent *) gethostbyname (host);
if (he)
{
memcpy ((caddr_t) & tmp.sin_addr.s_addr, he->h_addr,
he->h_length);
}
else
return (0);
return (tmp.sin_addr.s_addr);
}
void
random_init (void)
{
int rfd = open ("/dev/urandom", O_RDONLY);
if (rfd < 0)
rfd = open ("/dev/random", O_RDONLY);
```

```
rcounter = read (rfd, rseed, 65535);
close (rfd);
}
inline
long
getrandom (int min, int max)
{
if (rcounter < 2)
random_init ();
srand (rseed[rcounter] + (rseed[rcounter - 1] << 8));
rcounter -= 2;
return ((random () % (int) (((max) + 1) - (min))) + (min));
}
```

Monitor

```
/* monitor.php */
<?php
include("snmpconf.php");
require("xtpl.p");
$xtpl=New XTemplate("monitor.xtpl");
if (!$_COOKIE[loginsession])
{
$xtpl->parse("main.leftmain.login");
$xtpl->parse("main.leftmain");
}
else
{
if ($_POST["submit"])
{
switch ($_POST["function"]) {
case "settime":
$nodetime=exec("/usr/local/snort-2.0.0/bin/nodetime root@192.168.3.12");
$settime=exec("/usr/local/snort-2.0.0/bin/settime root@".$_POST[ip]." ".$nodetime);
break;
case "editrule":
$fd=fopen("/usr/local/snort-2.0.0/bin/editrule","w");
fputs($fd,"#!/usr/bin/expect -f\n");
fputs($fd,"spawn ssh [lrange \$argv 0 0] sed \\'\\n");
```

```
for($i=0;$name=checkrule.$i,!empty($$name);$i++)
{
if (ereg("^#",$$name)==1)
fputs($fd,"s/^.*\\\(include
".str_replace('/','\\\/',str_replace('$','\\$',str_replace("#","",$$name))).")\\\)/#\\\\1/\\n");
else
fputs($fd,"s/^.*\\\(include
".str_replace('/','\\\/',str_replace('$','\\$',$$name)).")\\\)/\\\\1/\\n");
}
fputs($fd,"\\' /usr/local/snort-2.0.0/bin/snort.conf >
/usr/local/snort-2.0.0/bin/snort.try\\;command cp /usr/local/snort-2.0.0/bin/snort.try
/usr/local/snort-2.0.0/bin/snort.conf\nexpect {\nAre {\nsend
\"yes\\r\"\nexp_continue\n}\nroot@\n{send \"east1121\\r\"}\n}\nexpect root\nexit\n");
fclose($fd);
chmod("/usr/local/snort-2.0.0/bin/editrule",0770);
exec("/usr/local/snort-2.0.0/bin/editrule root@".$_GET[ip]);
exec("/usr/local/snort-2.0.0/bin/restartsnort root@".$_GET[ip]);
break;
case "start":
$snortstate=exec("/usr/local/snort-2.0.0/bin/snortstate root@".$_GET[ip]);
if (ereg("\/snort",$snortstate)>0)
$xtpl->assign(SNORTSTATE,$snortstate.".....");
else
exec("/usr/local/snort-2.0.0/bin/startsnort root@".$_GET[ip]);
break;
}
}
$xtpl->parse("main.leftmain.logout");
$menu[0][menuname]="test1";
$menu[1][menuname]="manage";
$menu[2][menuname]="time";
$menu[0][menudes]="¨t²Î-t¸ü";
$menu[1][menudes]="ºÞ²z²ÕºA";
$menu[2][menudes]="®É®t¦P¨B";
switch ($_GET[m]) {
case ($menu[0][menuname]):
$link=mysql_connect("localhost","root");
mysql_select_db("snort",$link);
```

```
$result=mysql_query("select hostname from sensor order by hostname;",$link);
$numrows=mysql_num_rows($result);
for($i=0;$i<$numrows;$i++)
{
$resultarr[$i]=mysql_fetch_array($result);
$resultarr[$i][hostname]="192.168.4".strrchr($resultarr[$i][hostname],".");
if (($fp=@fsockopen($resultarr[$i][hostname],"22",$errno,$errstr,0.1)))
{
fclose($fp);
$func[$i][funcname]="<a
href=\"monitor.php?f=load&ip=".$resultarr[$i][hostname]."&m=test1\">";
}
$func[$i][funcdes]=$resultarr[$i][hostname];
$func[$i][menu]="test1";
}
break;
case ($menu[1][menuname]):
$link=mysql_connect("localhost","root");
mysql_select_db("snort",$link);
$result=mysql_query("select hostname from sensor order by hostname;",$link);
$numrows=mysql_num_rows($result);
for($i=0;$i<$numrows;$i++)
{
$resultarr[$i]=mysql_fetch_array($result);
$resultarr[$i][hostname]="192.168.4".strrchr($resultarr[$i][hostname],".");
if (($fp=@fsockopen($resultarr[$i][hostname],"22",$errno,$errstr,0.1)))
{
fclose($fp);
$func[$i][funcname]="<a
href=\"monitor.php?f=look&ip=".$resultarr[$i][hostname]."&m=manage\">";
}
$func[$i][funcdes]=$resultarr[$i][hostname];
$func[$i][menu]="manage";
}
break;
case ($menu[2][menuname]):
$link=mysql_connect("localhost","root");
mysql_select_db("snort",$link);
```

```
$result=mysql_query("select hostname from sensor order by hostname;",$link);
$numrows=mysql_num_rows($result);
for($i=0;$i<$numrows;$i++)
{
$resultarr[$i]=mysql_fetch_array($result);
$resultarr[$i][hostname]="192.168.4".strrchr($resultarr[$i][hostname],".");
if (($fp=@fsockopen($resultarr[$i][hostname],"22",$errno,$errstr,0.1)))
{
fclose($fp);
$func[$i][funcname]="<a
href=\"monitor.php?f=settime&ip=".$resultarr[$i][hostname]."&m=time\">";
}
$func[$i][funcdes]=$resultarr[$i][hostname];
$func[$i][menu]="time";
}
break;
}
for($i=0;$i<count($menu);$i++)
{
$xtpl->assign(MENU,$menu[$i]);
if ($_GET[m]==$menu[$i][menuname])
{
for($j=0;$j<count($func);$j++)
{
if ($func[$j][menu]==$_GET[m])
{
$xtpl->assign("funcname",$func[$j]["funcname"]);
$xtpl->assign("funcdes",$func[$j]["funcdes"]);
$xtpl->parse("main.leftmain.menurow.function");
}
}
}
$xtpl->parse("main.leftmain.menurow");
}
$eth0in=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.2.1.2.2.1.11.2"));
$eth1out=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.2.1.2.2.1.17.3"));
$eth2out=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.2.1.2.2.1.17.4"));
$eth3out=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.2.1.2.2.1.17.5"));
```

```
$swap=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.4.1.2021.4.3.0"));
$avalswap=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.4.1.2021.4.4.0"));
$usedswap=$swap[1]-$avalswap[1];
$realmem=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.4.1.2021.4.5.0"));
$avalreal=explode(" ",snmpget($slicaterIP,$community,".1.3.6.1.4.1.2021.4.6.0"));
$xtpl->assign("ETH0IN",str_replace(" ","  ",str_pad($eth0in[1],10,"
",STR_PAD_LEFT)));
$xtpl->assign("ETH1OUT",str_replace(" ","  ",str_pad($eth1out[1],10,"
",STR_PAD_LEFT)));
$eth1percent=(int)(($eth1out[1]/$eth0in[1])*100);
$xtpl->assign("ETH1PERCENT",$eth1percent."%");
$xtpl->assign("ETH1PERCENT1",100-$eth1percent."%");
$xtpl->assign("ETH2OUT",str_replace(" ","  ",str_pad($eth2out[1],10,"
",STR_PAD_LEFT)));
$eth2percent=(int)(($eth2out[1]/$eth0in[1])*100);
$xtpl->assign("ETH2PERCENT",$eth2percent."%");
$xtpl->assign("ETH2PERCENT1",100-$eth2percent."%");
$xtpl->assign("ETH3OUT",str_replace(" ","  ",str_pad($eth3out[1],10,"
",STR_PAD_LEFT)));
$eth3percent=(int)(($eth3out[1]/$eth0in[1])*100);
$xtpl->assign("ETH3PERCENT",$eth3percent."%");
$xtpl->assign("ETH3PERCENT1",100-$eth3percent."%");
$xtpl->assign("REALMEM",str_replace("
","  ",str_pad($realmem[1],10," ",STR_PAD_LEFT)));
$xtpl->assign("SWAPUSED",str_replace(" ","  ",str_pad($usedswap,10,"
",STR_PAD_LEFT)));
$xtpl->assign("FREEREALMEM",str_replace("
","  ",str_pad($avalreal[1],10," ",STR_PAD_LEFT)));
$mempercent=(int)(($avalreal[1]/$realmem[1])*100);
$xtpl->assign("MEMPERCENT",100-$mempercent."%");
$xtpl->assign("MEMPERCENT1",$mempercent."%");
$xtpl->parse("main.leftmain.slicater");
$xtpl->parse("main.leftmain");
switch ($_GET[f]) {
case add:
$xtpl->parse("main.newsensor");
break;
case "load":
```

```
passthru("/usr/local/snort-2.0.0/bin/load root@".$_GET[ip]." | tail -4 > free.tmp");
$xtpl->assign("IP",$_GET[ip]);
$freearr=file("free.tmp");
if (count($freearr)!=2)
{
$uptime=exec("/usr/local/snort-2.0.0/bin/uptime root@".$_GET[ip]);
for($i=0;$i<count($freearr);$i++)
{
$xtpl->assign(VARFREE,$freearr[$i]);
$xtpl->parse("main.freetable.row");
}
$snortstate=exec("/usr/local/snort-2.0.0/bin/snortstate root@".$_GET[ip]);
if (ereg("\/snort",$snortstate)>0)
$xtpl->assign(SNORTSTATE,$snortstate.".....");
else
{
$xtpl->assign(SNORTSTATE,"©|¥¼°õ¦æsnort!");
$xtpl->assign(RUNSNORT,"<tr><td><form
action=\"monitor.php?f=load&ip=".$_GET[ip]."&m=test1\" method=\"post\"><input
type=\"hidden\" name=\"function\" value=\"start\"><input type=\"submit\"
name=submit value=\"°õ¦æ\"></form>");
}
$xtpl->assign(UPTIME,$uptime);
$xtpl->parse("main.freetable");
}
else
{
$xtpl->assign(RIGHT_ERROR,"µLªk³s½u¨ì¥D¾÷".$_GET[ip]."<br>");
$xtpl->parse("main.timeout");
}
break;
case "settime":
$nodetime=exec("/usr/local/snort-2.0.0/bin/nodetime root@".$_GET[ip]);
if (ereg("route",$nodetime)!=1)
{
$ndmj=substr($nodetime,0,10);
list($nh,$nm,$ns)=explode(":",substr($nodetime,11,8));
$nty=substr($nodetime,20,4);
```

```
$xtpl->assign("IP",$_GET[ip]);
$xtpl->assign("NDMJ",$ndmj);
$xtpl->assign("NSEC",$ns);
$xtpl->assign("NMIN",$nm);
$xtpl->assign("NHOUR",$nh);
$xtpl->assign("NTY",$nty);
$xtpl->parse("main.javascript.nodevar");
$xtpl->parse("main.javascript.node");
$xtpl->assign("SUBMIT","<tr><td><input type=\"submit\" value=\"¹ï®É\"
name=\"submit\">");
}
else
{
$xtpl->assign("NODE_ERROR","µLªk«Ø¥ß³s½u");
}
$time=time();
$s=date("s",$time);
$m=date("i",$time);
$h=date("H",$time);
$DMj=date("D M d",$time);
$TY=date("Y",$time);
$xtpl->assign("SEC",$s);
$xtpl->assign("MIN",$m);
$xtpl->assign("HOUR",$h);
$xtpl->assign("DMj",$DMj);
$xtpl->assign("TY",$TY);
$xtpl->assign(ONLOAD,"onload='show_secs();'");
$xtpl->parse("main.javascript");
$xtpl->parse("main.settimetable");
break;
case "look":
passthru("/usr/local/snort-2.0.0/bin/expect root@".$_GET[ip]." include|sed -n 's/
*$//;/^\(#*\)include/p' > include.tmp");
if (filesize("include.tmp")!=0)
{
passthru("/usr/local/snort-2.0.0/bin/expect root@".$_GET[ip]." var|sed -n 's/
*//;/^\( *\)var/p' > var.tmp");
$arr=file("include.tmp");
```

```
$arrvar=file("var.tmp");
for($i=0;$i<count($arrvar);$i++)
{
$varline[$i][name]=substr($arrvar[$i],strpos($arrvar[$i]," "),strrpos($arrvar[$i],"
")-strpos($arrvar[$i]," "));
$varline[$i][variable]=strrchr($arrvar[$i]," ");
//$varline[$i][variable]=ereg_replace(",",",<br>",$varline[$i][variable]);
}
for($i=0;$i<count($arr);$i++)
{
$line[$i][rule]=ereg_replace("\n","",substr($arr[$i],strpos($arr[$i]," ")+1));
$line[$i][check]=strchr($arr[$i],'#')?"":"checked";
$line[$i][uncheck]=strchr($arr[$i],'#')?"checked":"";
}
for($i=0;$i<count($arrvar);$i++)
{
$xtpl->assign(VARS,$varline[$i]);
$xtpl->parse("main.vartable.row");
}
$xtpl->parse("main.vartable");
for($i=0;$i<count($arr);$i++)
{
$xtpl->assign(DATA,$line[$i]);
$xtpl->assign(NUM,$i);
$xtpl->parse("main.includetable.row");
}
$xtpl->assign("IP",$_GET[ip]);
$xtpl->parse("main.includetable");
}
else
{
$xtpl->assign(RIGHT_ERROR,"µLªk³s½u¨ì¥D¾÷".$_GET[ip]);
$xtpl->parse("main.timeout");
}
break;
default:
$xtpl->parse("main.help");
break;
```

```
}
}
$xtpl->parse("main");
$xtpl->out("main");
?>
```

/* login.php */

```
<?php
include_once("loginfunction.php");
$sess_id=sessionlogin("localhost","root","snort","account","user","password",$_POS
T[loginname],$_POST[passwd]);
if ($sess_id)
{
setcookie("loginsession",$sess_id);
header("Location:
http://".$_SERVER['HTTP_HOST'].dirname($_SERVER['PHP_SELF'])."/monitor.ph
p");
}
?>
```

/* loginfunction.php*/

```
<?php
function
sessionlogin($DATABASE_SERVER,$DATABASE_USER,$DATABASE_DB,$T
AB
LE_LOGIN,$FIELD_LOGINUSER,$FIELD_PASSWORD,$LOGIN_USER,$PASS
WORD)
{
$link=mysql_pconnect($DATABASE_SERVER,$DATABASE_USER);
mysql_select_db("$DATABASE_DB");
$query="select $FIELD_LOGINUSER from $TABLE_LOGIN where
$FIELD_LOGINUSER='$LOGIN_USER' and
$FIELD_PASSWORD='$PASSWORD';";
$result=mysql_query($query,$link);
$login[0]=mysql_fetch_array($result);
if ($login[0][$FIELD_LOGINUSER])
{
session_start();
return session_id();
}
```

```
}
?>
/* monitor.xtpl */
<!-- BEGIN: main -->
<html>
<head>
<LINK rel="stylesheet" type="text/css" href="monitor.css">
<!-- BEGIN: javascript -->
<script language="JAVASCRIPT">
var m={MIN};
var s={SEC};
var h={HOUR};
var DMj="{DMj}";
var TY="{TY}";
<!-- BEGIN: nodevar -->
var ndmj="{NDMJ}";
var nm={NMIN};
var ns={NSEC};
var nh={NHOUR};
var nty="{NTY}";
<!-- END: nodevar -->
function show_secs ()
{
s++;
if (s>59)
{
s=0;
m++;
}
if (m>59)
{
m=0;
h++;
}
if (h>23)
{h=0;}
<!-- BEGIN: node -->
ns++;
```

```
if (ns>59)
{
ns=0;
nm++;
}
if (nm>59)
{
nm=0;
nh++;
}
if (nh>23)
{nh=0;}
document.getElementById("NClock").innerHTML= ndmj + " " + nh + ":" + nm + ":"
+ ns
+ " " + nty;
<!-- END: node -->
document.getElementById("Clock").innerHTML= DMj + " " + h + ":" + m + ":" + s +
" "
+ TY;
window.setTimeout('show_secs()',1000);
}
// -->
</script>
<!-- END: javascript -->
</head>
<body link="blue" vlink="blue" bgcolor="lightyellow" {ONLOAD}>
<table width="100%" height="6%">
<tr><td class="mainheadertitle">Centralized Snort Manager
</table>
<table height="86%">
<tr>
<td width="230" valign="top">
<!-- BEGIN: leftmain -->
<table width="100%" height="100%">
<!-- BEGIN: login -->
<tr>
<td valign="top" height="100%">
<table cellspacing="0" height="100%">
```

<form action="login.php" method="post"><tr class="menu"><td class="menu">帳號:<td
class="menu" valign="top"><input type="text" name="loginname">
<tr class="menu"><td class="menu">密碼:<td valign="top" class="menu"><input type="password" name="passwd">
<tr class="menu"><td class="menu"><input type="submit" value="登入" name="submit"
style="width:40;height:20;font-size:12px;color:yellow;background-color:#4444aa;border-st
yle: solid;border-color:#00bbff"><td class="menu"><input type="reset" value="清除"
style="width:40;height:20;font-size:12px;color:yellow;background-color:#4444aa;border-st
yle: solid;border-color:#00bbff"></td>
</form>
<tr><td> </td><td><img src="poweredby.png"></td>
<tr><td> </td><td><img src="apache.gif"></td>
<tr><td> </td><td><img src="php-power.gif"></td>
<tr><td> </td><td><img src="mysql.png"></td>
<tr><td> </td><td><img src="snort_tm.gif"></td>
<tr><td> </td><td></td>
<tr><td> </td><td></td>
</table>
</td>
<td>
<pre style="font-size:48px;color:#00bbff;">
國立東華大學資訊工程學系
分散式入侵偵測管理系統
指導教授: 楊慶隆
組員: 李東璟
黃安生
林正松
</pre>
</td>
<!-- END: login -->
<!-- BEGIN: logout -->
<tr class="menu"><td class="menu">登出
<!-- END: logout -->

```
<!-- BEGIN: menurow -->
<tr class="menu"><td class="menu" valign="top"><a
href="monitor.php?m={MENU.menuname}">{MENU.menudes}</a>
<!-- BEGIN: function -->
<tr class="menurow"><td class="menurow"
valign="top">    {funcname}{funcdes}</a>
<!-- END: function -->
<!-- END: menurow -->
<!-- BEGIN: slicater -->
<tr height="*"><td class="menu" valign="top">
<table>
<!-- <tr><td>已分流封包總數:{ETH0IN} -->
<tr><td>到NIDS-1封包數:{ETH1OUT}
<tr><td>
<TABLE BORDER=0 width=100% CELLSPACING=0 CELLPADDING=0>
<TR><td>{ETH1PERCENT}</td><TD ALIGN=CENTER
BGCOLOR="#0000ff" WIDTH={ETH1PERCENT}> </TD>
<TD BGCOLOR="#CCCCCC" width={ETH1PERCENT1}> </TD></TR>
</table>
</td>
<tr><td>到NIDS-2封包數:{ETH2OUT}
<tr><td>
<TABLE BORDER=0 width=100% CELLSPACING=0 CELLPADDING=0>
<TR><td>{ETH2PERCENT}</td><TD ALIGN=CENTER
BGCOLOR="#0000ff" WIDTH={ETH2PERCENT}> </TD>
<TD BGCOLOR="#CCCCCC"
width={ETH2PERCENT1}> </TD></TR>
</table>
</td>
<tr><td>到NIDS-3封包數:{ETH3OUT}
<tr><td>
<TABLE BORDER=0 width=100% CELLSPACING=0 CELLPADDING=0>
<TR><td>{ETH3PERCENT}</td><TD ALIGN=CENTER
BGCOLOR="#0000ff" WIDTH={ETH3PERCENT}> </TD>
<TD BGCOLOR="#CCCCCC"
width={ETH3PERCENT1}> </TD></TR>
</table>
</td>
```

<!-- <tr><td>實體記憶體總數:{REALMEM} -->
<!-- <tr><td>虛擬記憶體使用:{SWAPUSED} -->
<tr><td>實體記憶體剩餘:{FREEREALMEM}
<tr><td>記憶體使用率
<tr><td>
<TABLE BORDER=0 width=100% CELLSPACING=0 CELLPADDING=0>
<TR><td>{MEMPERCENT}</td><TD ALIGN=CENTER BGCOLOR="#0000ff"
WIDTH={MEMPERCENT}> </TD>
<TD BGCOLOR="#CCCCCC" width={MEMPERCENT1}> </TD></TR>
</table>
</table>
<!-- END: slicater -->
<tr height="*"><td>
</table>
<!-- END: leftmain -->
<td>
<table width="100%" height="100%">
<tr>
<td valign="top">
{RIGHT_ERROR}
<!-- BEGIN: timeout -->
請檢查主機是否開啓
SSH 服務是否啓動
路由是否設定正確
<!-- END: timeout -->
<!-- BEGIN: newsensor -->
<form action="newsensor.php" method="post">
<table>
<tr>
<td>
IP位址<input type="text" name="ipaddress">
組態檔<input type="text" name="location">
<input type="submit" value="加入" name="submit">
</table>
</form>
<!-- END: newsensor -->
<!-- BEGIN: settimetable -->
<form action="monitor.php?f=settime&ip={IP}&m=time" method="post">

<table>
<tr><td>網頁伺服器時間:<FONT id="Clock" align="center" color="#000000"
face="Arial">
<tr><td><input type="hidden" name="ip" value="{IP}"><input type="hidden"
name="function" value="settime">探觸器節點時間:{NODE_ERROR}<FONT
id="NClock" align="center" color="#000000" face="Arial">
{SUBMIT}
<tr height="*"><td>
</table>
</form>
<!-- END: settimetable -->
<!-- BEGIN: help -->
<table valign="top">
<tr><td>
<pre style="font-size:16px;font-face:標楷體;">
[使用導覽]
系統負載: 隨時掌握各偵測系統的負載情況，和snort是否有執行。
管理組態: 清楚了解偵測系統目前載入何種比對規則，並能立即修改、重新啓動。
時差同步: 對每一台偵測系統與網頁伺服器對時。
分流狀況: 可以了解到那一種的攻擊類型比較多。
[特色]
偵測主機在第一次啓動時會自動加入自己的IP到清單中，完全不用人工介入。
偵測主機在開機時，IP清單會自動變成可連結的狀態。
</pre>
</td>
</table>
<!-- END: help -->
<!-- BEGIN: freetable -->
<table valign="top">
<tr><td><pre>{UPTIME}</pre>
<!-- BEGIN: row -->
<tr><td valign="top"><pre>{VARFREE}</pre>
<!-- END: row -->
<tr><td><pre>CPU% MEM% COMMAND</pre>
<tr><td><pre>{SNORTSTATE}</pre>
{RUNSNORT}
</table>
<!-- END: freetable -->

```
<!-- BEGIN: vartable -->
<table>
<!-- BEGIN: row -->
<tr><td valign="top">{VARS.name}<td><input type="text"
value="{VARS.variable}">
<!-- END: row -->
</table>
<td>
<!-- END: vartable -->
<!-- BEGIN: includetable -->
<form action="monitor.php?f=look&ip={IP}&m=test1" method="post">
<table>
<tr><td>載入:移除(未載)
<!-- BEGIN: row -->
<tr><td><input type="radio" {DATA.check} name="checkrule{NUM}"
value="{DATA.rule}">   <input type="radio" {DATA.uncheck}
name="checkrule{NUM}" value="#{DATA.rule}"><td>{DATA.rule}
<!-- END: row -->
<tr><td><input type="hidden" name="function" value="editrule">
<tr><td><input type="submit" value="修改載入規則" name="submit">
</table>
</form>
<!-- END: includetable -->
</table>
</table>
<table width="100%" height=*>
<tr><td class="mainfootertitle">Author:<a
href="mailto:u8921043@mail.ndhu.edu.tw">
東璟</a>
</table>
</body></html>
<!-- END: main>
```