國 立 交 通 大 學

電信工程學系碩士班

碩 士 論 文

利用列表顯示線上封包統計
連續測試分散式阻斷服務攻擊

Sequential testing with tabulated online packet statistics for DDoS Detection

研 究 生：吳孟諭

指導教授：李程輝　教授

中華民國九十四年六月

利用列表顯示線上封包統計
連續測試分散式阻斷服務攻擊

Sequential testing with tabulated online packet statistics for DDoS Detection

研 究 生：吳孟諭　　Student：Meng-Yu Wu

指導教授：李程輝　教授　　Advisor：Prof. Tsern-Huei Lee

國立交通大學

電信工程學系碩士班

碩士論文

A Thesis
Submitted to Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Communication Engineering
June 2005
Hsinchu, Taiwan, Republic of China.

中 華 民 國 九 十 四 年 六 月

利用列表顯示線上封包統計
連續測試分散式阻斷服務攻擊

研究生：吳孟諭　　指導教授：李程輝　教授

國立交通大學電信工程學系碩士班

中文摘要

在這篇論文裡，我們提出一種有效率偵測與過濾阻斷服務攻擊的方法。 我們的系統叫 STTOPS（Sequential Testing with Tabulated Online Packet Statistics for DDoS Detection）能在一個緊密，固定尺寸的架構裡使用有效的探索法監控許多網路位址。 我們證明與 TOPS 相比較，STTOPS 在一個標準基準數據集方面有更高的平均準確和更低的平均誤報警能發現阻斷服務攻擊。與 TOPS 相比較，STTOPS 的關鍵好處是它使用較少的計算資源，而且在攻擊期間不會減慢下來。

Sequential testing with tabulated online packets statistics for DDoS Detection

Student：Meng-Yu Wu　　Advisor：Prof. Tsern-Huei Lee

Department of Communication Engineering

National Chiao-Tung University

Abstract

In this thesis, we present an efficient method for detecting and filtering denial-of-service bandwidth attacks. Our system called STTOPS （Sequential Testing with Tabulated Online Packet Statistics for DDoS Detection） can monitor a large number of network addresses in a compact, fixed-size structure using several effective heuristics. We demonstrate that STTOPS can detect bandwidth attacks in a standard benchmark dataset with a higher average accuracy and a lower average false alarms than TOPS. A key benefit of STTOPS is that it uses less computational resources than TOPS and does not slow down during an attack.

誌謝

能完成這篇論文，首先我最想感謝的是我的指導教授李程輝老師，老師使我真正知道什麼才是做學問的態度和精神。他總是能夠適時的引導我到正確的方向上，就像是研究道路上的一盞明燈，讓初窺學術研究領域的我不至於迷失方向。老師是一個真正的學者，他在研究過程中對學問的嚴謹和尊重，使我深深折服和欽佩。我祝福老師，希望他身體健康，萬事如意。同時我也要感謝蔣名駿和林冠亨同學對我寫程式上的幫忙，不厭其煩的讓我詢問。

目錄

表目錄

圖目錄

第一章

簡介

隨著科技的演變，電腦和網路已逐漸成為人與人之間溝通的主要工具，隨著使用者數目與需求的因素，網路頻寬不斷的增加，從早期的數據機(56kbits/sec)上網到現在用ＡＳＤＬ(2Mbits/sec)為主流，可見網路成長的力道之強勁。隨著網路的發展伴隨而來的是網路安全這個重要的課題，很多使用者藉由電腦與網路彼此溝通，但是由於現有的網路架構與規範不足，致使很多分散式阻斷服務攻擊、駭客入侵、蠕蟲感染等各式各樣網路安全事件層出不窮，導致現在很多專家、學者全力投入研究網路安全這個重要的課題。

網路安全的問題又以分散式阻斷服務攻擊最為聞名，依照攻擊原理分成兩類：(1) 頻寬攻擊 (2) 系統資源攻擊。頻寬攻擊的原理是利用分散在各地的電腦主機，針對網路鏈結 (network link) 送大量的垃圾封包塞滿它的頻寬，當網路鏈結的頻寬被垃圾封包佔滿時，就會導致使用者無法連線而達到阻斷服務的目的。系統資源攻擊的原理是利用分散在各地的電腦主機，針對某一台電腦主機與它建立許多半開的連線 (half-open connection)，導致正常的使用者無法與它建立連線而達到阻斷服務的目的。兩者的攻擊原理相似，但是攻擊對象不同，衍伸出的問題解決方式也不相同。就頻寬攻擊而言，在攻擊者端由於攻擊的流量不大，不易偵測到流量的異常，但是經由網路慢慢匯集到受害者端時，攻擊流量就相當驚人，導致受害的網路連線無法處理那麼多封包，產生進出邊緣路由器(edge router)網路流量不對等的現象發生。經由邊緣路由器進入受害者網路的流量很大，但出去受害者網路的流量相對的很小，我們就利用此項特性來偵測頻寬攻擊。就系統資源攻擊而言，攻擊者端會與受害者端建立許多半開的連線，受害者端可以縮短等待連線建立的時間、限制建立半開連線的數目等方式來因應，有趣

的是攻擊者端送出許多建立連線的 SYN 封包，但是攻擊封包填上造假的源位址，導致 ACK 封包無法送回到攻擊者端，這也存在進出邊緣路由器封包不對等的現象，也可利用上述的偵測方式去偵測。

許多有關處理分散式阻斷服務攻擊的方法已經被提出。入口過濾（Ingress filtering)[4]試著在攻擊者端檢查源位址是否符合子網路位址相符，如果不符合就認定它有攻擊企圖而將它丟棄。另一種替代的方法是利用啟發規則（heuristic rule）來分析區域網路和偵測可惜封包。舉例來說，基於歷史過濾(history-based filtering)[9]在攻擊期間，會根據過去記載的源位址來丟棄之前沒見過的源位址封包。它所根據的原理是在攻擊期間，許多新的源位址都是經過造假用來做攻擊用途。在這篇論文當中，我們著眼於之前提過的進出網路流量不對等，這項重大特徵是由 Gil 和 Polleto 兩位先生在 MULTOPS（Multi-Level Tree for Online Packet Statistics）[2]這篇論文中所提出。後來又有 Samuel Abdeisayed*、 David Glimsholt*、 Christopher Leckie、 Simon Ryan 和 Samer Shami*提出 TOPS（tabulated online packet statistics）[1]加以改進。TOPS 這篇論文最主要的貢獻是提出新穎的資料結構，但偵測攻擊方式則有相當的複雜度。另外在蠕蟲偵測方面，Stuart E. Schechter、Jaeyeon Jung 和 Arthur W. Berger 提出 Sequential hypothesis testing [3]，利用條件機率漸進式的推斷是否遭受蠕蟲攻擊，這個概念相當簡單且實用。我們在本篇論文中將會結合 TOPS 與 Sequential hypothesis testing 兩種概念的優點，大大簡化分散式阻斷服務攻擊的偵測方式。

第二章

相關成果

在上述簡介中我們曾提及 MULTOPS、TOPS 與 Sequential hypothesis testing 這三種偵測方法，我們在以下段落中，分別依序介紹它們的方法與優缺點。

2.1 線上封包統計之多層樹狀資料結構

首先，要介紹的機制是 MULTOPS，它是一個樹狀的資料結構，裡面每個節點都包含了子網路字首（subnet prefix）的封包速率統計資料且依照不同聚集層級(aggregation level)而有所不同。見下圖 2.1。

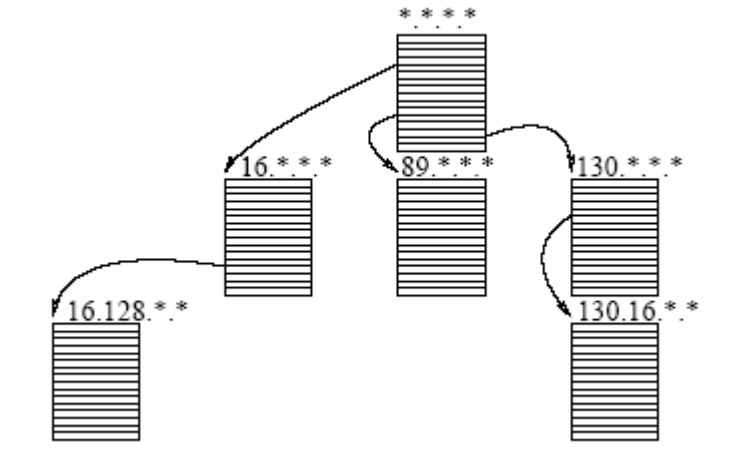

圖 2.1 MULTOPS

MULTOPS 是根據位址分為四個層級，它的範圍包含了所有 IPv4 位址。在樹狀資料結構裡的每個節點都包含了 256 筆資料，每筆資料包含了 to-rate、from-rate 和一個指標指向下一個層級的節點。依照樹的層級不同而有 0-bit、

8-bit、16-bit 與 24-bit 字首的差別，越高層級的節點就包含了從它以下所有節點的 to-rate、from-rate 聚集得總和。

MULTOPS 有個很重要的前提就是假定在正常運作下，從 A 到 B 的封包速率是正比於從 B 到 A 的封包速率。利用這個特性將機制為攻擊者模式和受害者模式，在攻擊者模式下 from-rate 除以 to-rate 若低於 Rmin(threshold)，就往下長出新的節點繼續監控，若是聚集的 from-rate 除以 to-rate 沒有低於 Rmin 就放棄自它以下的節點，利用這樣動態管理的方式進行網路流量的監控。受害模式亦然，差別在於 from-rate 除以 to-rate 要高於 Rmax(threshold)才會往下增生新的節點。見圖 2.2。

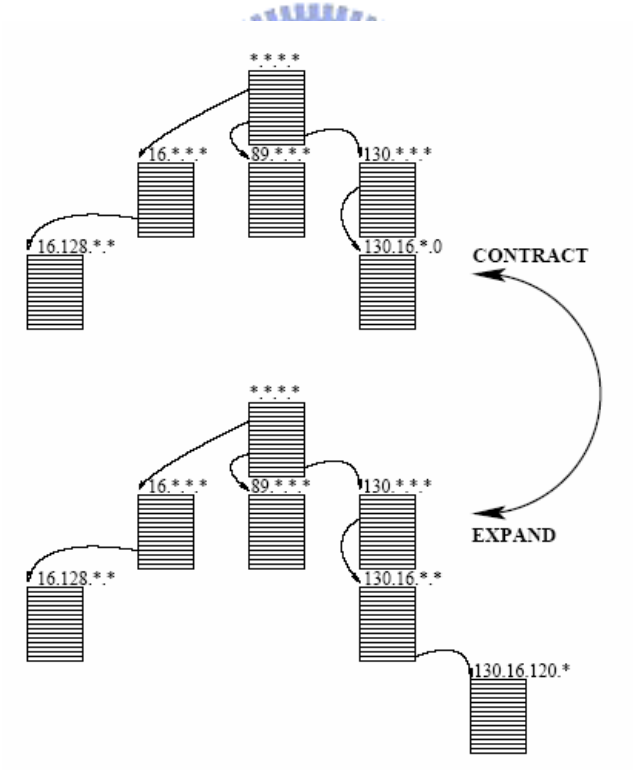

圖 2.2 MULTOPS expansion and contraction

此種方法的優點是，他看出了分散式阻斷服務攻擊進出流量不對等的特性。最大的缺點在於，動態管理樹狀資料結構不是一件容易的事，特別是在高速網路

下會造成 CPU 龐大的運算負擔，所以衍生出 TOPS 改良來這項機制。

2.2 線上表列式封包統計

TOPS (Tabulated Online Packet Statistics)是放置於邊緣路由器上(leaf router)，採用固定大小的資料結構來監控網路流量的異常，見下圖 2.3。

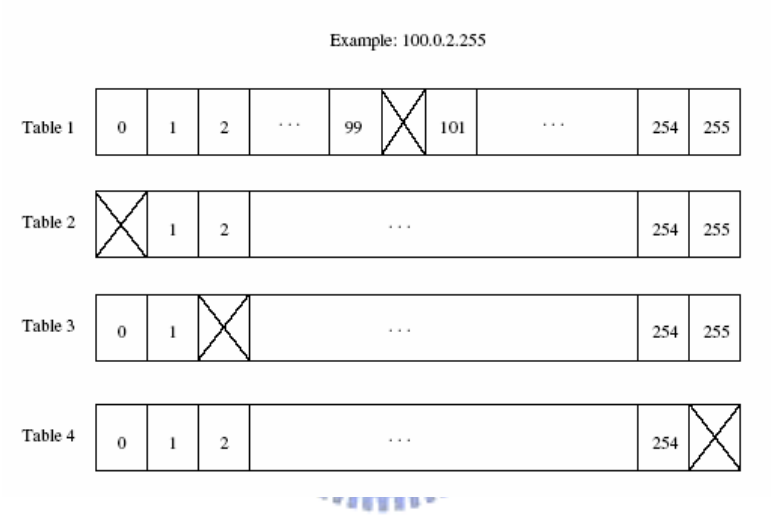

圖 2.3 Example of TOPS table showing entries for IP address 100.0.2.255

它依據 IP 位址的特性(ex.140.113.13.112)建立四個大小含有 256 個格子(entry)的表格，每個格子裡都有兩個參數分別記載進來(Pin)與出去(Pout)子網路的封包流量，再依照 Pin/Pout 的比值去判斷是否遭受攻擊，所利用的偵測原理與 MULTOPS 相似。它最大的貢獻在於有新穎的資料結構提供儲存統計資料，大大減低原本 MULTOPS 的動態運算量，使得路由器在遭受攻擊的情況下，也不會耗損太多的運算資源。

為了提高發出警報的準確性，加入了確定門檻(certainty threshold)，這

是個不易實現的方法,必須要根據進來或出去目標網路的封包速率產生一個累積機率分布,再根據累積分布判斷是否為遭受攻擊。要儲存此項資訊實為不易,是實現 TOPS 最大的困難。

2.3 連續假設試驗

在蠕蟲偵測方面,善意的電腦主機只會和遠端系統,試著建立應該會建立成功的連線,但是受到蠕蟲感染的電腦主機,則是會盡一切所能感染其他的電腦主機,但是因為有些電腦它的某個連接埠沒開,甚至是電腦主機沒開機,導致在一定時間內很多的連線是失敗的,這個現象跟 TCP SYN Flooding 很像,差別在於受到蠕蟲感染的電腦主機是一台對多台電腦主機進行感染,而 TCP SYN Flooding 則是多台電腦主機攻擊一台電腦主機,衍伸出來的偵測位置則有差異。遭受蠕蟲感染的電腦主機在發動攻擊時,會在它所經過的邊緣路由器就被偵測到,而發動 TCP SYN Flooding 攻擊的意圖,則是在遠處受害者端的邊緣路由器才被偵測到。但是兩者的偵測原理都是一樣的,在一定時間內若是建立連線失敗的次數太多,我們就會懷疑是否發生發生攻擊。以下我們就簡略介紹 Sequential hypothesis testing 的偵測方式。

設定 Yi 是一個代表第 i 次連線結果的隨機變數。如果連線成功,Yi=1;如果連線失敗,Yi=0。又設定 H1 是某台電腦主機在從事攻擊的前提,H0 是沒有在從事攻擊。假設以 Hj 為前提之下,隨機變數 Yi|Hj,i=1,2,3,…,彼此是獨立同分佈(independent and identically distribution)。換句話說,在以某個條件為前提之下,任意兩個想建立連線的嘗試都會有相同的可能性成功,他們成功的機會也彼此不相關。我們表示白努力隨機變數 Yi 的分布如下:

$$\Pr[Y_i = 0|H_0] = \theta_0, \quad \Pr[Y_i = 1|H_0] = 1 - \theta_0$$
$$\Pr[Y_i = 0|H_1] = \theta_1, \quad \Pr[Y_i = 1|H_1] = 1 - \theta_1$$

給定善意主機建立連線的成功機率比被蠕蟲感染的主機還高，$\theta 0 > \theta 1$。

Sequential hypothesis testing 是藉著一連串發生的事件 Yn≡（Y1,Y2,…Yn）來判斷是否遭受攻擊。藉著產生一個比率Λ(Yn)，分子是一連串事件 Yn 在 H0 前提下的條件機率，分母是一連串事件 Yn 在 H1 前提下的條件機率。Λ(Yn)表示如下：

$$\Lambda(\mathbf{Y}_n) \equiv \frac{\Pr[\mathbf{Y}_n|H_1]}{\Pr[\mathbf{Y}_n|H_0]}$$

由於任意兩個前提下的機率分佈為獨立同分佈，我們可以將上式在分解成下式：

$$\Lambda(\mathbf{Y}_n) \equiv \prod_{i=1}^{n} \frac{\Pr[Y_i|H_1]}{\Pr[Y_i|H_0]}$$

假設第 i 次的觀察ϕ(Yi)，表示如下：

$$\phi(Y_i) \equiv \frac{\Pr[Y_i|H_1]}{\Pr[Y_i|H_0]} = \begin{cases} \frac{\theta_1}{\theta_0} & \text{if } Y_i = 0 \text{ (success)} \\ \frac{1-\theta_1}{1-\theta_0} & \text{if } Y_i = 1 \text{ (failure)} \end{cases}$$

由上式可知，當建立連線成功時，ϕ(Yi)＞1；當建立連線失敗時，ϕ(Yi)＞1。設定上限$\eta 1$，超過上限時我們選定 H1 也就是遭受攻擊。設定下限$\eta 0$，低於下

限時我們選定 H0 也就是沒有遭受攻擊。示意圖如下：

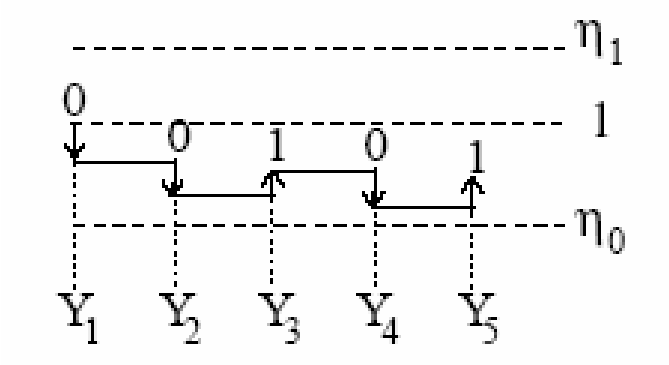

圖 2.4 Sequential hypothesis testing concept plot

Sequential hypothesis testing 的概念由此就可以看出來，藉著觀察一段時間所發生的事件去判定實際的狀況，我們將運用這個概念在我們的架構中去簡化 TOPS 的運算，結合兩者概念的優點做出混合式分散阻斷服務攻擊偵測。

第三章

提出的演算法

在第二章相關成果當中，我們提及了 TOPS 與 Sequential hypothesis testing，這兩個方法各有優點，我們要把它們各自的優點取出成為另一個機制。我們試著討論在偵測分散式阻斷服務攻擊時，第一、要儲存哪些資訊，第二、要用什麼方式儲存才會精簡，第三、儲存的資料要如何運用來取得判斷遭受攻擊的依據。我們會在以下的段落文字裡詳細的探討，並提出一套完整的偵測架構 STTOPS（Sequential Testing with Tabulated Online Packet Statistics)。

首先，我們擬採用 TOPS 的儲存架構去存取進出邊緣路由器的封包數(Pin、Pout over some interval t)，假設採用受害者模式(victim mode)，TOPS 是利用 R=Pin/Pout 的比值來判斷是否發生異常，若 R＞Rmax，則發出警告(alarm)，再經過之前所儲存的流量累積機率分布去判定是否發生攻擊(attack)，困難點在於要儲存累積機率分布不是一件容易的事，所以我們將利用 sequential hypothesis testing 的概念來簡化所需的運算。一樣針對每個 table entry 儲存兩個變數 Pin、Pout，再加上兩個變數 A(A 初始值設為 0)與 Y(Y 初始值設為 0)。在之前 TOPS 是利用比值 R=Pin/Pout＞Rmax，在此我們轉換成 D＝Pin－Rmax＊Pout 簡化計算之複雜度，若 D≧0，則發出警告(alarm)、把 Y 加一，若 D＜0，則視為正常，把 Y 減一。每次更改 Y 值後，把 Y 與我們事先設定好的門檻(threshold)α做比較。若 Y≧α，則判定為攻擊狀態(attack)，A 設為 1； Y 若大於α，則 Y 設定為α，是因為不想讓 Y 無限制的往上升，導致前面的攻擊偵測會影響到後面的攻擊偵測。若 0≦Y＜α，則 Y、A 值不變。在判定單一 entry 是否為攻擊或正常狀態之前，我們不對可疑封包採取任何的措施，一旦某個 IP 位址有四個的 table entries 判定為攻擊狀態(k=4)，則採取限流(throttle)等

相關因應措施。

以下的流程圖是因應 MIT Lincoln Lab 1999 DARPA data set 所做的修改，把 t 改為讀入的封包數 n。每讀進 n 個封包數就去做一次判斷是否異常，並比較 Y 的值是否有超過上限 α。換而言之，我們從流量觀察中去推斷目前子網路的流量是否正常或是遭受到攻擊。下圖 3.1 是混合式阻斷服務攻擊偵測的流程圖。

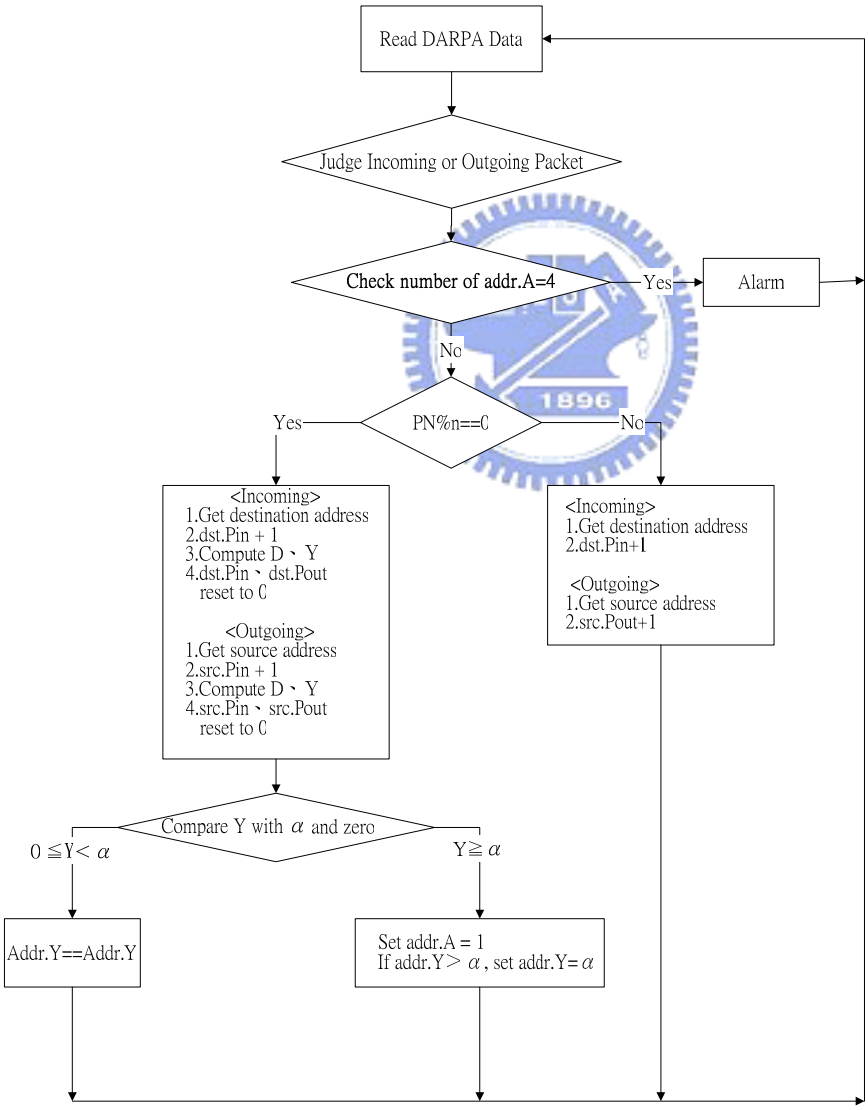

圖 3.1 STTOPS Dos Detection flow chart

一開始先從 MIT Lincoln Lab 1999 DARPA data set 提供的 tcpdump 檔案裡讀出一個封包資料，利用 IP 標頭裡的目的地位址資料來判斷此封包是進入或出去子網路，再藉著路由器上統計資料來判斷此封包的目的地位址是否遭受攻擊，如果遭受攻擊就發出警告，如果沒有遭受攻擊，就進行進出封包流量統計。直到讀入的封包數達到 n 時，就計算全部表格的 Y 值且判斷是否超過上限α，並把進入封包數（Pin）與出去封包數（Pout）歸零重新計數。反覆經過數次統計，若 Y$\geqq\alpha$，判定為遭受攻擊並把 A 設為 1，如果 Y$\geqq\alpha$，把 Y 設定為α。若 0$\leqq$Y$<\alpha$，則保留 Y、A 的值讓之後的統計再去判斷。

第四章

模擬結果

在這一章，我們將介紹由 MIT Lincoln Lab 所製作的 1999 DARPA off-line intrusion data set[6]，它主要是包含了入侵資料(intrusion data set)，但是也有若干阻斷服務攻擊資料包含在裡面，我們的模擬是使用這些資料去做測試。在我們的模擬中，我們是利用第四和第五個禮拜的資料去做測試，最後跟 TOPS 所列出的模擬結果做比較。

4.1 1999 DARPA evaluation Data Set

在 1999 DARPA off-line intrusion data set 裡，第四和第五個禮拜包含了一些阻斷服務攻擊的資料在裡面，它的攻擊分成許多類型，我們的目標設定在偵測外部電腦主機對特定監視的網域所做的攻擊與特定監視網域遭受攻擊時進出不對等流量特性的正確性。從外部電腦主機發動的攻擊分為兩種類型：(1)TCP SYN floods (2)ICMP floods。進一步檢視 TCP SYN floods 發現大多數的 SYN 封包都會被受害者承認(acknowledged)。因此這種類型的低強度 TCP SYN floods 將不會被偵測出來，因為進出流量不對等的比重太低，所以我們就專注屬於 ICMP floods 類型名字叫做 pod 與 smurf。在 TOPS 這篇論文把 pod 與 smurf 當成是必須偵測出的攻擊，而其他的阻斷服務攻擊偵測當成是額外的優勢。下圖 4.1 為 1999 年模擬的網路拓圖。

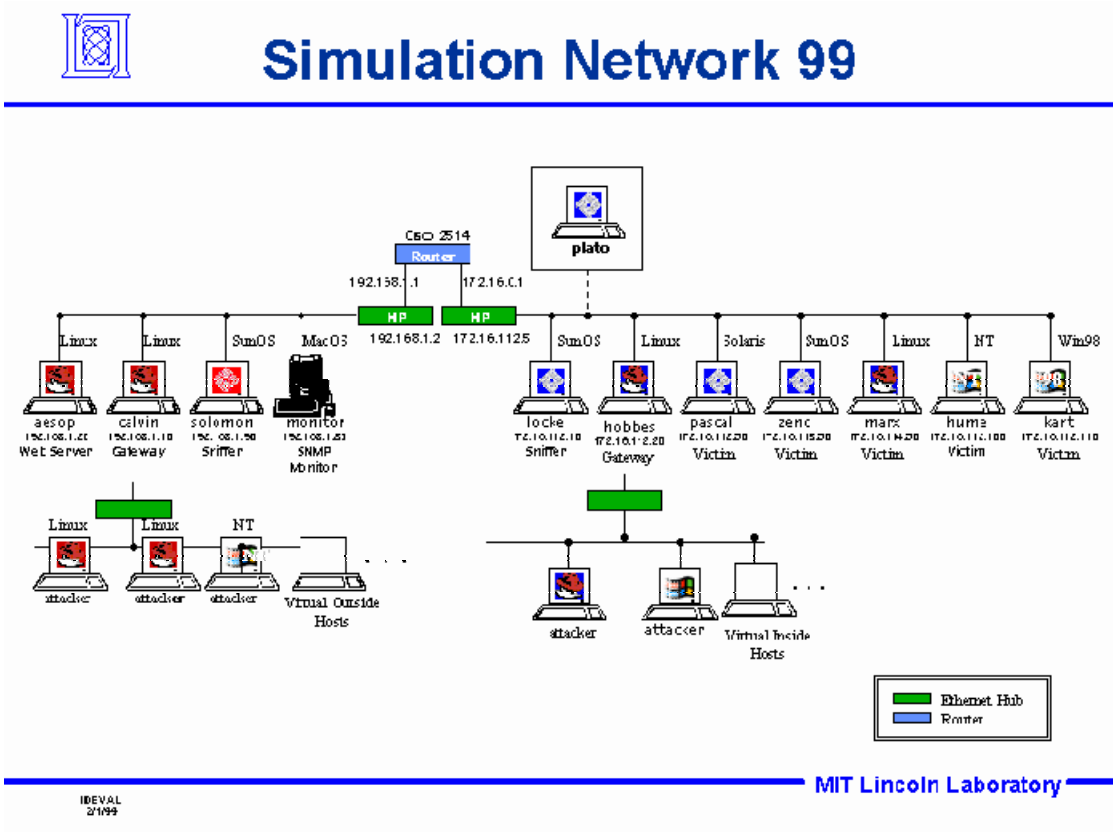

圖 4.1 1999 Simulation Network Topology

以下的表格是根據 TOPS 所拿來做測試資料的日期，再依照 MIT Lincoln Lab 所提供的詳細攻擊資料，製作成表格以方便有需求的人可以使用。其中 pod 與 smurf 攻擊都用粗體顯示以方便辨識。

表 4.1：1999 Week 4 Monday attack information

Week	Day	Attack	Time	Duration	Attacker	Victim
4	Mon	ps	08:18:35	00:46:05	209.154.98.104	172.16.112.50
4	Mon	sendmail	08:48:12	00:00:02	202.49.244.10	172.16.114.50
4	Mon	ntfsdos	09:08:00	00:08:00	172.16.112.100	172.16.112.100
4	Mon	portsweep	09:15:05	00:20:32	172.16.118.20	172.16.113.50
4	Mon	sshtrojanInstall	09:36:23	00:05:23	202.77.162.213	172.16.114.50
4	Mon	portsweep	11:15:15	00:04:01	172.16.118.50	192.168.1.1
4	Mon	xsnoop	11:22:43	00:01:17	128.223.199.68	172.16.114.168
4	Mon	snmpget	11:45:12	00:33:31	204.97.153.43	172.16.0.1
4	Mon	guesstelnet	11:47:16	00:00:10	192.5.41.239	172.16.113.50
4	Mon	portsweep	12:22:15	00:05:01	153.107.252.61	172.16.112.100
4	Mon	guessftp	13:33:16	00:00:22	172.16.118.70	172.16.112.50
4	Mon	ftpwrite	13:58:16	00:05:45	194.27.251.21	172.16.112.50
4	Mon	yaga	15:50:15	04:13:06	172.16.118.70	172.16.112.100
4	Mon	crashiis	16:13:08	00:00:05	172.16.118.70	172.16.112.100
4	Mon	portsweep	16:27:10	00:04:56	202.077.162.213	172.16.114.169
4	Mon	secret	18:24:19	00:08:41	195.115.218.108	172.16.112.50
4	Mon	secret	18:24:19	00:08:41	195.115.218.108	172.16.112.50
4	Mon	smurf	21:34:16	00:00:11	6.238.105.108	172.16.112.50

表 4.2：1999 Week 4 Wednesday attack information

Week	Day	Attack	Time	Duration	Attacker	Victim
4	Wen	satan	08:04:14	00:00:13	209.30.70.14	172.16.114.50
4	Wen	netcat	09:26:12	00:00:04	207.230.54.203	172.16.112.100
4	Wen	imap	09:38:13	00:00:53	208.254.251.132	172.16.114.50
4	Wen	ppmarcro	10:14:36	00:13:01	194.27.251.21	172.16.112.100
4	Wen	processtable	10:13:18	00:28:02	172.16.118.60	172.16.113.50
4	Wen	fdformat	10:38:00	00:03:00	console	172.16.112.50
4	Wen	netcat	11:11:08	00:00:32	206.48.44.18	172.16.112.100
4	Wen	warezmaster	11:25:39	00:00:47	202.027.121.118	172.16.112.50
4	Wen	arppoison	11:30:13	00:11:21	172.16.118.20	172.16.113.50
4	Wen	ncftp	12:12:38	00:01:29	152.169.215.104	172.16.114.50
4	Wen	secret	12:28:15	00:08:12	196.37.75.158	172.16.112.50
4	Wen	named	13:00:32	00:05:10	194.7.248.153	172.16.112.20
4	Wen	guessftp	13:42:16	00:01:56	208.240.124.83	172.16.113.50
4	Wen	smurf	18:29:25	00:00:01	1.12.120.6	172.16.112.100
4	Wen	guest	15:53:15	00:02:43	209.12.13.144	172.16.112.50
4	Wen	portsweep	16:43:15	00:03:54	208.240.124.83	172.16.112.100
4	Wen	mailbomb	16:54:17	00:03:01	194.7.248.153	172.16.112.50
4	Wen	guesstelnet	17:58:17	00:03:48	209.1.12.46	172.16.112.50
4	Wen	snmpget	19:12:16	00:16:57	207.181.92.211	172.16.0.1

表 4.3：1999 Week 4 Friday attack information

Week	Day	Attack	Time	Duration	Attacker	Victim
4	Fri	smurf	08:45:18	00:00:02	1.12.120.6	172.16.112.50
4	Fri	arppoison	09:00:10	00:12:51	206.47.98.151	172.16.114.50
4	Fri	sshtrojan	09:55:15	00:10:29	195.73.151.50	172.16.114.50
4	Fri	ipsweep	10:03:10	00:15:01	172.16.112.5 172.16.112.10 194.7.248.153	172.16.112.1-10 194.7.248.153
4	Fri	xlock	10:42:04	00:01:19	139.134.61.42	172.16.114.50
4	Fri	named	10:53:24	00:03:35	194.7.248.153	172.16.112.20
4	Fri	portsweep	11:10:15	00:04:31	208.240.124.83	172.16.113.50
4	Fri	ncftp	11:45:13	00:01:29	172.16.118.70	172.16.114.50
4	Fri	netbus	11:51:00	00:04:13	209.1.12.46	172.16.112.100
4	Fri	mailbomb	12:32:17	00:12:59	202.72.1.77	172.16.113.50
4	Fri	named	13:49:30	00:00:58	195.73.151.50	172.16.112.20
4	Fri	ipsweep	14:00:15	00:08:39	128.223.199.68 172.16.114.50 204.71.51.16 204.233.47.21 207.114.237.57 209.1.12.46	172.16.114.1 172.16.114.3-5 172.16.114.50 204.233.47.21
4	Fri	loadmodule	16:21:16	00:05:41	194.27.251.21	172.16.113.50
4	Fri	sechole	16:50:15	00:45:11	195.115.218.108	172.16.112.100
4	Fri	portsweep	18:10:15	00:00:05	202.49.244.10	172.16.113.50
4	Fri	ipsweep	19:25:14	00:42:18	172.16.112.5	172.16.112.1-254

					172.16.112.10 172.16.112.50 172.16.112.100 172.16.112.194 172.16.112.207 172.16.113.105 172.16.114.207 194.7.248.153	194.7.248.153
4	Fri	secret	20:34:00	00:02:00	console	172.16.112.50

表 4.4：1999 Week 5 Monday attack information

Week	Day	Attack	Time	Duration	Attacker	Victim
5	Mon	**pod**	08:39:52	00:00:10	202.77.162.213	172.16.112.50
5	Mon	portsweep	08:43:17	00:03:53	172.16.118.10	192.168.1.1
5	Mon	**pod**	08:48:33	00:00:31	202.77.162.213	172.16.114.50
5	Mon	warezclient	08:59:16	00:00:41	207.75.239.115	172.16.112.50
5	Mon	**smurf**	09:33:00	00:02:00	login.session.x.x	172.16.112.50
5	Mon	portsweep	09:43:11	00:03:43	208.240.124.83	172.16.112.50
5	Mon	apache2	10:29:22	00:17:37	202.77.162.213	172.16.114.50
5	Mon	guesstelnet	10:58:14	00:03:20	192.5.41.239	172.16.118.80
5	Mon	dosnuke	11:45:27	00:01:33	172.16.115.234	172.16.112.100
5	Mon	loadmodule	12:03:14	00:11:15	172.16.114.207	172.16.113.50
5	Mon	ffbconfig	12:11:18	00:12:28	135.13.216.191	172.16.112.50
5	Mon	**smurf**	13:18:12	00:00:01	23.234.78.52	172.16.114.50
5	Mon	arppoison	13:30:14	00:14:37	152.169.215.104	172.16.112.100
5	Mon	apache2	14:05:43	00:11:09	152.169.215.104	172.16.114.50
5	Mon	**pod**	14:22:30	00:00:01	10.11.22.33	172.16.113.50
5	Mon	imap	14:46:19	00:00:10	172.16.117.103	172.16.114.50
5	Mon	ipsweep	15:00:16	00:15:21	128.223.199.68 172.16.113.50 204.71.51.016 204.233.47.21 207.114.237.57 209.1.12.46	172.16.113.1 172.16.113.3-5 172.16.113.50 204.233.47.21
5	Mon	dict	16:32:17	00:16:35	172.16.118.10	172.16.114.50

5	Mon	syslogd	17:19:10	00:00:01	172.5.3.5	172.16.112.50
5	Mon	neptune	18:04:04	00:06:51	10.20.30.40	172.16.112.50
5	Mon	crashiis	18:36:11	00:00:07	202.72.1.77	172.16.112.100
5	Mon	ls	18:57:21	00:00:01	195.73.151.50	172.16.112.20
5	Mon	dosnuke	19:48:01	00:01:41	206.48.44.18	172.16.115.234
5	Mon	udpstorm	20:00:27	00:15:00	172.16.112.50	172.16.113.*
5	Mon	selfping	20:17:12	00:00:03	135.13.216.191	172.16.112.50
5	Mon	ncftp	20:46:13	00:01:29	172.16.118.70	172.16.114.50

表 4.5：1999 Week 5 Tuesday attack information

Week	Day	Attack	Time	Duration	Attacker	Victim
5	Tue	tcpreset	08:11:15	00:10:50	135.8.60.182	172.16.112.50
5	Tue	teardrop	08:32:14	00:00:01	207.230.54.203	172.16.114.50
5	Tue	casesen	08:53:17	00:49:38	172.16.113.204	172.16.112.100
5	Tue	xsnoop	09:19:01	00:01:12	194.7.248.153	172.16.112.50
5	Tue	selfping	09:45:13	00:00:03	192.182.91.233	172.16.112.50
5	Tue	xterm1	10:07:18	00:40:43	152.169.215.104	172.16.112.194
5	Tue	ftpwrite	10:19:16	00:14:04	172.16.114.207	172.16.112.50
5	Tue	back	11:31:21	00:20:38	206.48.44.50	172.16.114.50
5	Tue	ps	11:20:09	01:40:51	199.227.99.125	172.16.112.50
5	Tue	neptune	11:38:04	00:13:41	10.20.30.40	172.16.114.50
5	Tue	httptunnel	12:06:32	00:03:30	172.16.112.50	196.37.075.158
5	Tue	eject	12:55:14	00:16:32	152.169.215.104	172.16.112.50
5	Tue	pod	13:06:00	00:00:30	166.102.114.43	172.16.113.50
5	Tue	yaga	13:28:11	00:42:19	194.7.248.153	172.16.112.100
5	Tue	crashiis	13:50:03	00:00:05	194.7.248.153	172.16.112.100
5	Tue	ppmarcro	14:26:26	00:12:52	199.174.194.16	172.16.112.100
5	Tue	syslogd	14:13:56	00:00:01	172.3.45.1	172.16.112.50
5	Tue	perl	14:24:17	00:14:47	207.103.80.104	172.16.114.207
5	Tue	fdformat	16:24:15	01:00:00	196.38.75.158	172.16.112.50
5	Tue	queso	16:54:20	00:00:51	199.227.99.125	172.16.113.50
5	Tue	neptune	18:16:05	00:03:26	10.20.30.40	192.168.1.1
5	Tue	dosnuke	20:57:03	00:01:33	172.16.115.234	172.16.112.100
5	Tue	ipsweep	21:15:54	00:19:30	195.73.151.50	172.16.0.1

						172.16.112.10 172.16.112.20 172.16.112.50 172.16.112.100 172.16.112.149 172.16.112.194 172.16.112.200 172.16.112.207 172.16.113.50 172.16.113.84 172.16.113.105 172.16.113.204 172.16.114.50 172.16.114.148 172.16.114.168 172.16.114.169 172.16.114.207 172.16.115.5 172.16.115.87 172.16.115.234 172.16.116.44 172.16.116.194 172.16.116.201 172.16.117.52 172.16.117.103

						172.16.117.111 172.16.117.132 172.16.118.10 172.16.118.20 172.16.118.30 172.16.118.40 172.16.118.50 172.16.118.60 172.16.118.70 172.16.118.80 172.16.118.90 172.16.118.100
5	Tue	ncftp	21:45:02	00:01:28	172.16.118.20	172.16.114.50
5	Tue	udpstorm	05:30:38	00:00:01	172.16.112.50	172.16.113.*

表 4.6：1999 Week 5 Thursday attack information

Week	Day	Attack	Time	Duration	Attacker	Victim
5	Thurs	ps	08:33:00	00:03:00	console	172.16.112.50
5	Thurs	phf	09:01:08	00:31:46	206.48.44.50	172.16.114.50
5	Thurs	casesen	09:16:20	02:03:47	172.16.112.149	172.16.112.100
5	Thurs	ntfsdos	10:21:00	00:15:00	172.16.112.100	172.16.112.100
5	Thurs	portsweep	10:34:11	00:05:44	153.10.8.174	172.16.112.50
5	Thurs	ntinfoscan	11:26:37	00:16:09	206.48.44.18	172.16.112.100
5	Thurs	yaga	11:52:05	00:15:39	194.7.248.153	172.16.112.100
5	Thurs	crashiis	11:57:01	00:00:03	194.7.248.153	172.16.112.100
5	Thurs	httptunnel	12:06:30	00:03:31	172.16.112.50	196.37.75.158
5	Thurs	fdformat	12:57:17	00:40:10	194.27.251.21	172.16.112.50
5	Thurs	satan	14:58:29	00:02:12	209.74.60.168	172.16.114.50
5	Thurs	teardrop	15:53:18	00:00:01	199.227.99.125	172.16.114.50
5	Thurs	sechole	16:03:15	00:21:56	208.240.124.83	172.16.112.100
5	Thurs	resetscan	17:01:19	00:01:02	172.16.117.103	172.16.112.*
5	Thurs	ipsweep	17:16:10	00:01:31	172.16.112.5 172.16.112.10 207.136.086.223	172.16.112.1~10 207.136.86.223
5	Thurs	snmpget	17:50:12	00:05:06	194.27.251.21	172.16.0.1
5	Thurs	ntinfoscan	18:31:00	00:16:11	206.48.44.18	172.16.112.100
5	Thurs	ls	19:08:31	00:00:01	209.12.13.144	172.16.112.20
5	Thurs	warezclient	19:41:14	00:00:41	209.30.70.14	172.16.112.50
5	Thurs	mscan	19:58:59	04:03:05	207.136.86.223	172.16.112.10 172.16.112.20

						172.16.112.50 172.16.112.100 172.16.112.149 172.16.112.194 172.16.112.207 172.16.113.50 172.16.113.84 172.16.113.105 172.16.113.204 172.16.114.50 172.16.114.148 172.16.114.168-169 172.16.114.207 172.16.115.5 172.16.115.87 172.16.115.234 172.16.116.44 172.16.116.194 172.16.116.201 172.16.117.52 172.16.117.103 172.16.117.111 172.16.117.132 172.16.118.10

						172.16.118.20 172.16.118.30 172.16.118.40 172.16.118.50 172.16.118.60 172.16.118.70 172.16.118.80 172.16.118.90 172.16.118.100
5	Thurs	arppoison	22:51:14	00:28:01	135.13.216.191	172.16.112.100

4.2 比較線上表列式封包統計模擬結果

在 TOPS 是取禮拜三的測試資料當作是展示成果，我們也依循這個腳步跟它做個比較。下面表格 4.7 是簡略地將禮拜三的 smurf 攻擊做個表示，在表七右上欄位中 Packet Number 是指在 TCP dump file 裡 smurf 攻擊從開始到結束的封包編號，也就是說攻擊從編號 1,570,448 這個封包開始一直到編號 1,575,591 結束，這中間幾乎是攻擊的流量。

表 4.7：1999 Week 4 Wednesday

Attack Name	Duration	Attacker	Victim	Packet Number
smurf	00:00:01	1.12.120.6	172.16.112.100	1,570,448~1,575,591

再來就是參數設定的問題了，分別探討 k、n、Rmax 與 α 對實驗結果的影響。首先針對 K 的值來做探討，在 TOPS 論文中它設定 K=3 時，即可將攻擊封包做處理。而本論文是設定 K=4，也就是說要完全符合被攻擊的位址才將封包做處理。下表 4.8 為實驗結果，它的參數設定為 n=200、α=9。

表 4.8：K 值對偵測率與錯誤警告的影響

Attack packet detection rate（%）		
	K=4	K=3
Rmax=2	60.63	60.63
Rmax=4	60.63	60.63
Rmax=6	60.63	60.63
Rmax=8	60.63	60.63
Rmax=10	60.63	60.63
Rmax=20	60.63	60.63
False alarms		
	K=4	K=3
Rmax=2	610	614
Rmax=4	270	274
Rmax=6	0	1
Rmax=8	0	1
Rmax=10	0	1
Rmax=20	0	1

由上表 4.8 可知道 K 值對實際攻擊的偵測率不會造成影響，但是會影響錯誤警告，因為當 K=3 時，子網路內的某台主機遭受攻擊會連帶影響其他主機的封包也一併被丟棄，造成錯誤警告增加。

再來就 n 值的大小來探討它對模擬的影響。下表 4.9 為不同的 n 值對偵測率和錯誤警告的影響。它的參數設定為 K=4、α=9。

表 4.9：n 值對偵測率與錯誤警告的影響

Attack packet detection rate（%）			
	n=50	n=100	n=200
Rmax=2	90.85	80.85	60.63
Rmax=4	89.73	80.85	60.63
Rmax=6	89.73	80.85	60.63
Rmax=8	89.73	80.85	60.63
Rmax=10	89.73	80.85	60.63
Rmax=20	89.73	78.6	60.63
False alarms			
Rmax=2	1831	1326	610
Rmax=4	1371	989	270
Rmax=6	750	0	0
Rmax=8	0	0	0
Rmax=10	0	0	0
Rmax=20	0	0	0

由上表 4.9 得知 n 值越小越能快速偵測到攻擊發生，但相對的錯誤警告會增加。藉著提升 Rmax，我們可以消除錯誤警告的負面效果。在 n=50、Rmax=6 條件下會產生 750 個錯誤警告，因為當時的流量分布是 8 個 http 封包才有一個 ack 封包，又加上 n=50 使得快速累積到攻擊門檻。

最後探討 α 對對模擬的影響。下表 4.10 為不同的 α 值對偵測率和錯誤警告的影響。它的參數設定為 K=4、n=50、Rmax=6。

表 4.10：α 值對偵測率與錯誤警告的影響

α	1	2	3	4	5
Attack packet detection rate（%）	X	97.6	96.47	95.35	94.22
False alarms	X	1879	1295	1106	1022
α	6	7	8	9	10
Attack packet detection rate（%）	93.1	91.98	90.85	89.73	88.6
False alarms	921	836	793	750	622
α	11	12	13	14	15
Attack packet detection rate（%）	87.48	86.36	85.24	84.11	82.99
False alarms	576	495	452	409	364

由上表 4.10 得知 α 值對偵測率與錯誤警告的影響很小，隨著 α 值變大，錯誤警告會降低但偵測率也相對下降。圖 4.2、4.3 分別為 α 值對偵測率的影響和 α 值對錯誤警告的影響。

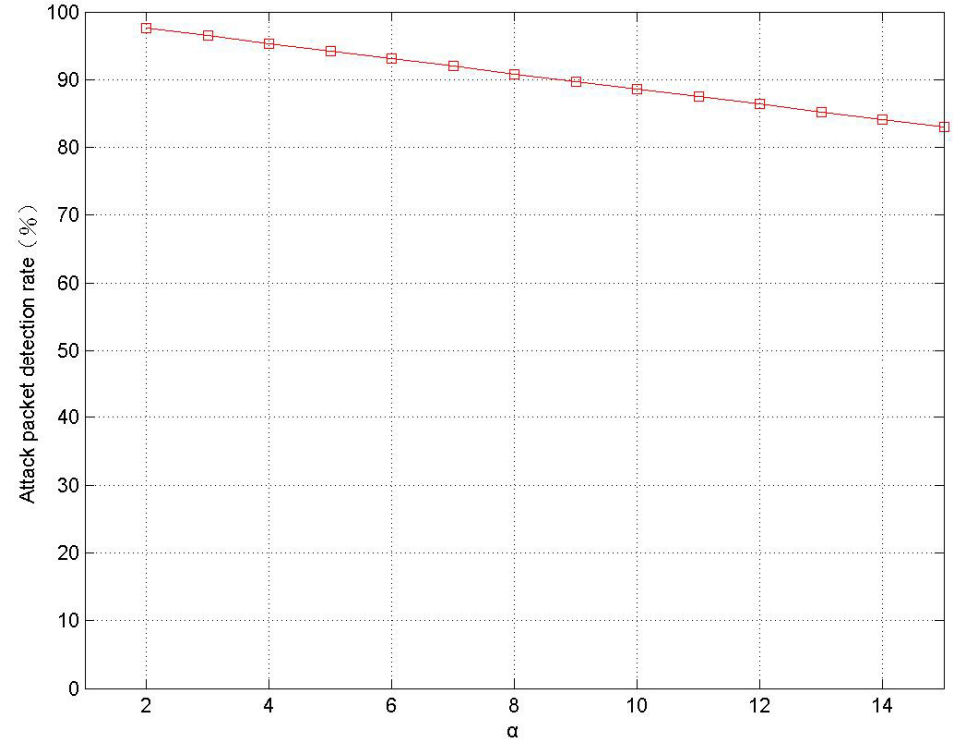

圖 4.2 α 值對偵測率的影響

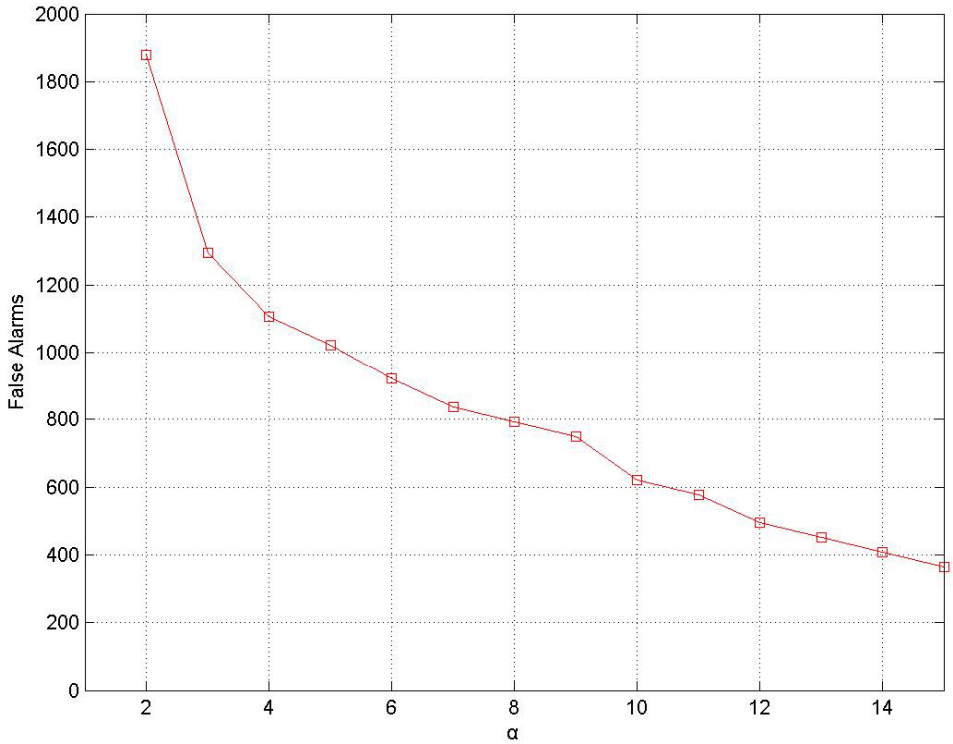

圖 4.3 α 值對錯誤警告的影響

由以上的各種實驗我們得出一個重要的結論，想要快速偵側攻擊發生則 n 值越小越好，但相對的誤判機率就會升高，此時須將 Rmax 值調高來減少誤判。我們建議各個參數值的設定為 K=4、n=50、Rmax≧8、α ≦10。

下圖 4.4 是 TOPS 把 Real alarm 與 False alarm 對 Rmax 做圖，去找出 Rmax 與 Real alarm、False alarm 的關係。測試的資料仍是 1999 Week 4 Wednesday。下圖 4.5 是我們提出的混合式偵測方法，把 Real alarm 與 False alarm 對 Rmax 畫出的曲線圖。

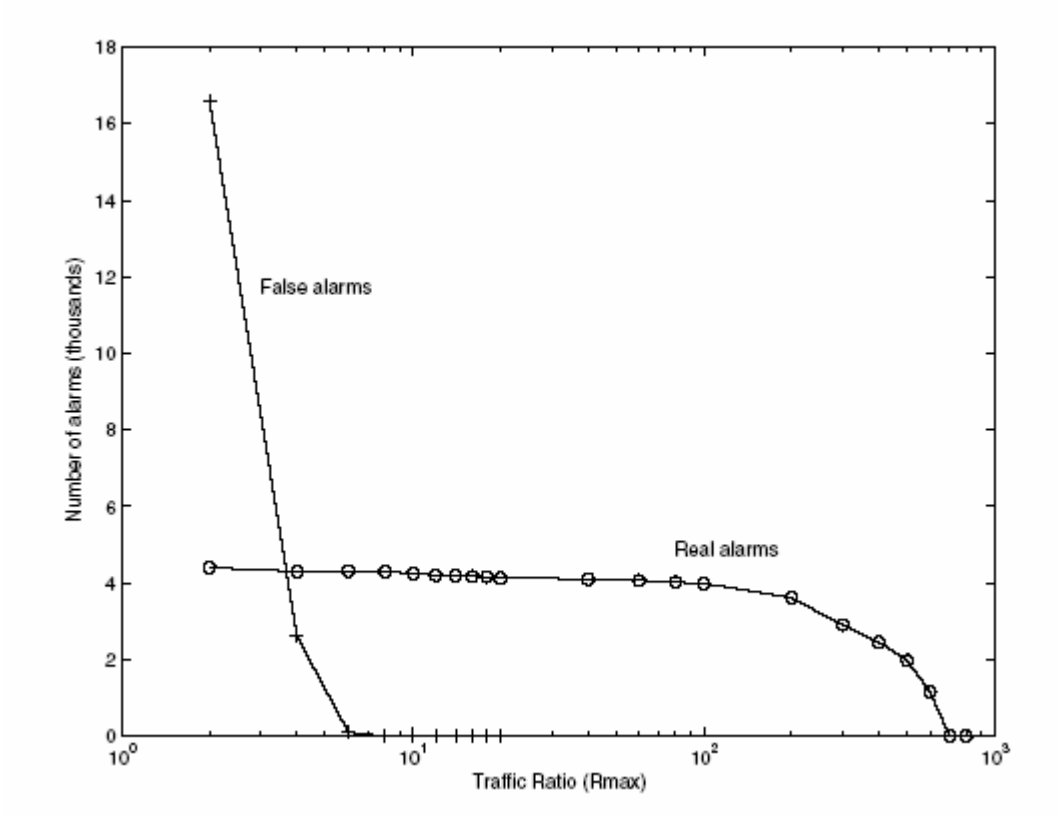

圖 4.4 TOPS 1999 Week 4 Wednesday Rmax versus Number of alarms

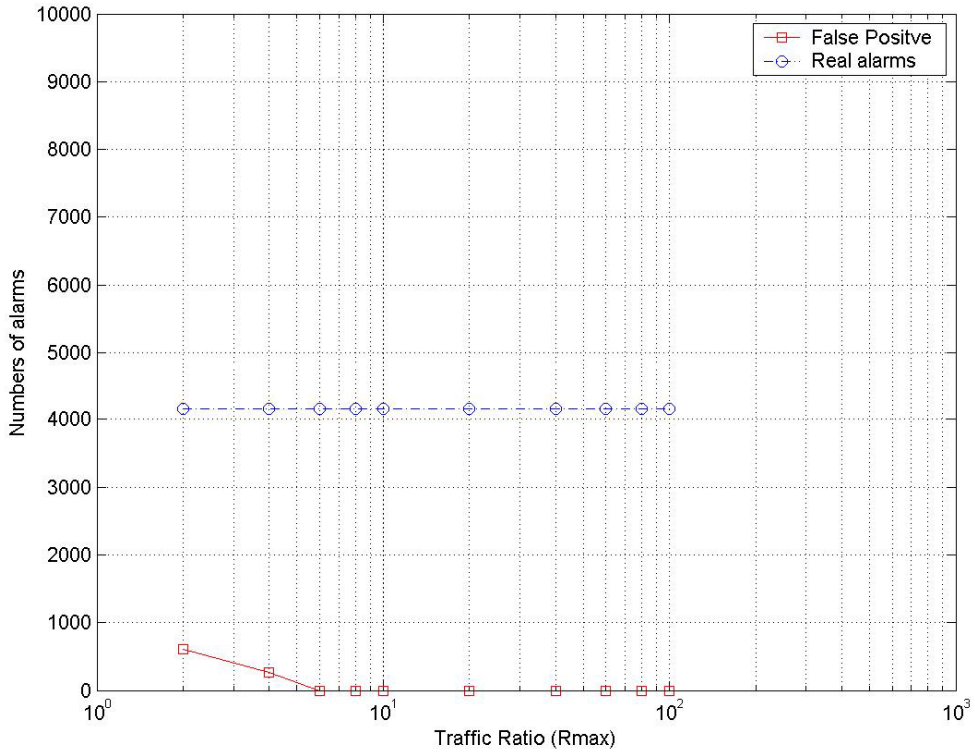

圖 4.5 STTOPS Detection 1999 Week 4 Wednesday Rmax versus Number of alarms

在圖 4.5 裡面我們可以清楚的看見不管流量比率是多少，我們都可以抓到攻擊發生，但是隨著流量比率的上昇，我們可以清楚的看到 False alarms 的急速減少。整體來說我們的混合式偵測方法平均的錯誤警告數遠小於 TOPS。

根據上面的模擬結果，我們定義攻擊封包偵測率、錯誤警告如下。以下的“攻擊”是指 pod 和 smurf。

【定義】

攻擊封包偵測率＝ 偵測出的攻擊封包數目÷ 實際發生的攻擊封包數目

錯誤警告＝ 除了實際發生攻擊封包之外的警告

由以上的定義我們可以把所有的模擬結果作成表格跟 TOPS 做比較，發現我們的結果平均來說比 TOPS 略勝一籌，顯示出我們的偵測方式既簡單就有效。

表 4.11：STTOPS DoS Detection vs TOPS

Week	Date	Attack Packet Detection Rate	
		STTOPS	TOPS
4	Mon	99.3%	99.5%
4	Wed	89.7%	81.2%
4	Fri	89.5%	95%
5	Mon	88%	37.6%
5	Tue	0%	0%

上表 4.11 Week 5 Tue 的偵測率為 0，因為當天只有 pod 攻擊沒有 smurf 攻擊，而本論文的方法是由進出的封包量來偵測攻擊，所以一旦遇到 pod 便束手無策。下圖 4.6 為 STTOPS 與 TOPS 的攻擊封包偵測率比較圖。

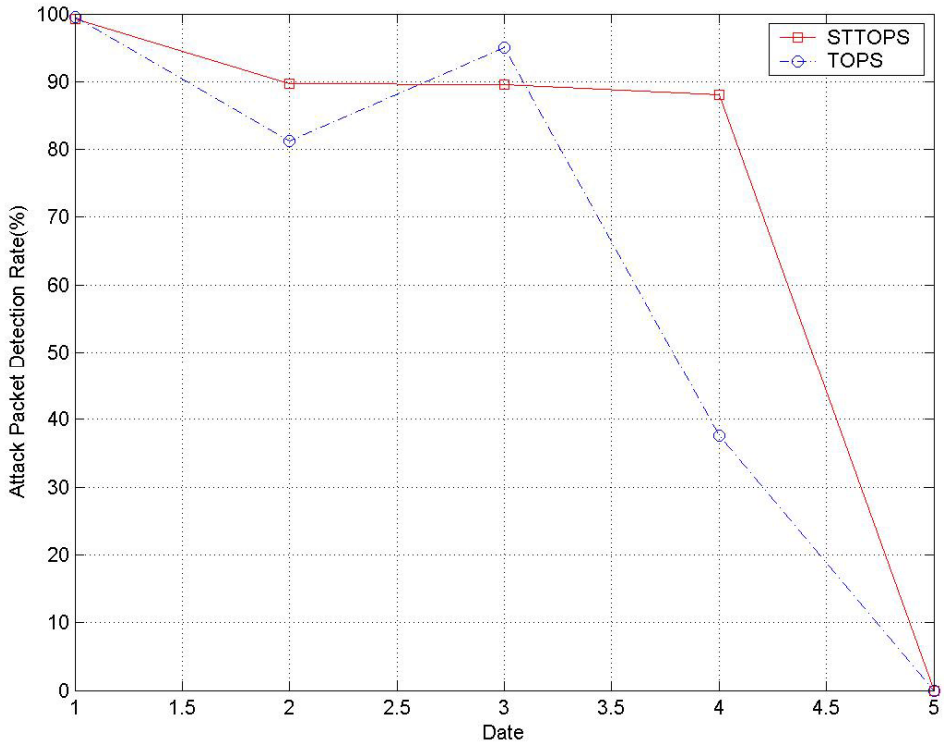

圖 4.6 STTOPS 與 TOPS 的攻擊封包偵測率比較圖

第五章

結論

我們在阻斷服務攻擊偵測方面，結合 TOPS 的儲存架構與 Sequential testing 的概念，使得平均偵測效果比原本的 TOPS 還好，演算法的實現與運算上也更容易。未來希望可以持續改進演算法，將演算法實現在硬體上，藉著實際的封包測試來看整體的表現。

參考文獻

[1]Samuel Abdeisayed*, David Glimsholt*, Christopher Leckie, Simon Ryan and Samer Shami*. “An Efficient Filter foe Denial-of-Service Bandwidth Attacks.” In Proceedings of IEEE GLOBALCOM 2003.

[2]T. Gil and M. Poletto. “MULTOPS：a data structure for bandwidth attack detection.” In Proceedings of the 10th USENIX Security Symposium, August 2001, Washington D.C., USA.

[3]Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishan. “Fast Portscan Detection Using Sequential Hypothesis Testing.”

[4]P. Ferguson and D. Senie. “Network Ingress Filtering：Defeating Denial of Service Attacks which employ IP Source Address Spoofing.” RFC2827, May 2000, http://www.ietf.org/rfc/rfc2827.txt

[5] J. Ioannidis and S. Bellovin. “Implementing Pushback：Router-Based Defense Against DDoS Attacks.” In Proceedings of the Network and Distributed System Security Symposium（NDSS 2002）, February 2002.

[6]R. Lippmann, et al. “Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation.” In Proceedings of the third International Workshop on Recent Advances in Intrusion Detection（RAID 2000）, pp. 162-182.

[7]K. Park and H. Lee "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack." In Proceedings IEEE INFOCOM 2001, April 2001, Anchorage, Alaska, USA, Vol.1, pp.338-347.

[8]T. Peng, C. Leckie and R. Kotagiri. "Adjusted Probabilistic Packet Marking for IP Traceback." In Proceedings of the Second IFIP Networking Conference (Network 2002),19-24 May 2002, Pisa, Italy.

[9]T. Peng, C. Leckie and R. Kotagiri. "Protection from Distributed Denial of Service Attack Using History-based IP Filtering." To appear in the IEEE International Conference on Communications (ICC 2003),11-15 May 2003, Anchorage, Alaska, USA.

[10]S.Savage, D. Wetherall, A. Karlin and T. Anderson. "Network support IP traceback." In IEEE/ACM Transaction on Networking, Vol.9 No.3, June 2001, pp.226-237.

[11]D. Song and A. Perrig. "Advanced and authenticated marking schemes for IP traceback." In Proceedings of IEEE INFOCOM 2001, April 2001, Anchorage, Alaska, USA, Vol.2, pp.878-886.

[12]S. Bellovin. The icmp traceback message. Internet Draft, IETF, March 2000. draft-bellovin-itrace-05.txt (work in progress). http://www.reseach.att.com/~smb.

[13]S. Felix Wu, Lixia Zhang, Dan Massey, and Allison Mankin. Intension-Driven ICMP Trace-Back. Internet Draft, IETF, February 2001. draft-wu-itrace-intension-00.txt (work in progress).

[14]Fabiano Dalpiaz. "Distributed Denial of Service attacks and defensive techniques" July 8, 2004.

[15]Matthew V. Mahoney and Philip K. Chan. "Learning Rules for Anomaly Detection of Hostile Network Traffic." In Proceedings of the third IEEE International Conference on Data mining (ICDM' 03)

附錄

簡歷

一. 個人資料：

姓名	吳孟諭
性別	男
生日	民國 69 年 10月 12日
出生地	台灣-嘉義市
身高	178cm
體重	70kg
血型	O
星座	天秤座
生肖	猴
婚姻	未婚
現職	國立交通大學電信工程學系 系統組 碩士班二年級
永久地址	608 嘉義縣水上鄉水頭村22鄰中興路79號
通訊地址	300 新竹市交通大學工程四館823實驗室
電話	03-5712121轉54570
E-Mail	u8813057.cm88@nctu.edu.tw keysea@banyan.cm.nctu.edu.tw

二. 學歷：

學　　校	系　　所	學　位	時 間 起 訖
1.嘉義高級中學		高中/畢業	84年9月－87年6月
2.國立交通大學	電信工程學系	學士/畢業	88年9月－92年6月
3.國立交通大學	電信工程學系 系統組	碩士/畢業	92年9月－94年6月