基於多代理人架構之洪水式分散阻絕服務攻擊偵防機制

劉經緯*
雲林科技大學資管所 研究生
Email： g9523802@yuntech.edu.tw

曹偉駿
大葉大學資訊管理系 副教授
wjtsaur@yahoo.com.tw

摘要

隨著網際網路的蓬勃發展下，造成網路攻擊日益頻繁，並且在攻擊手法推陳出新下，入侵手法亦趨向複雜化。由於惡意的分散式阻絕服務攻擊，可使得線上交易停擺，造成企業損失慘重，故建立有效率的預防性防禦機制是現今各企業首要目標。

綜觀目前抵抗洪水式分散阻絕服務攻擊是無法抵擋合法使用者攻擊，以及入侵偵測系統癱瘓時並無法即時備援，因此本研究將以多代理人架構，整合三種防禦機制。第一種是植基於橢圓曲線公開金鑰密碼系統以驗證是否為合法使用者，第二為轉移服務埠之防禦機制，以確保即使受到大量封包攻擊時，服務仍然可以正常運行。最後為備援機制，當代理人偵測入侵偵測系統受到分散式阻絕服務攻擊導致癱瘓時，將啟動備援機制，通知鄰近入侵偵測系統準備備援，如此一來可結合上述兩種防護而真正達到防禦功能。

最後，本研究亦開發出實際系統並模擬測試當主機受攻擊時，仍然可以正常提供服務，以及模擬當主機癱瘓時具備即時備援能力，以實現真正防禦洪水式分散阻絕服務攻擊。

關鍵字：網路安全，分散式阻絕服務攻擊，入侵偵測系統，橢圓曲線公開金鑰密碼系統，備援機制

Abstract

With the rapid growth of Internet, malicious attacks are getting more numerous and menacing. Distributed denial-of-service (DDoS) attacks are different from most other attacks, because they are not targeted at gaining access to information systems. These attacks focus on making a service unavailable for normal use, which is typically accomplished by exhausting some resource limitation on the network or within an operating system or application. This attack interferes with trading online and causes the damages to the business. Therefore, establishing efficient detecting and preventing schemes will become the main concern for the business.

General flood DDoS detecting schemes can't prevent attacks by legal users, and it can't backup immediately crashed servers, either. Therefore, this thesis proposes a multi-agent structure to integrate three protection schemes. The first is based on elliptic curve public key cryptosystems to authenticate users. The second is the protection scheme of the service port transformation, which servers still can operate normally even if being attacked by numerous abnormal packets. The third is the backup scheme. When the agent finds out flood DDoS attacks crash the servers, the backup scheme will start on to notice the near hosts to backup the crashed host.

In summary, this study is based on three schemes to develop a practical system, which still can provide services normally and also have the ability to backup hosts immediately even if they are attacked by flood DDoS.

1. 緒論

網際網路的盛行之下帶動了電子商務蓬勃發展，然而熱門網站相繼遇到駭客(hacker)攻擊，例如 2000 年的 Yahoo、ebay.com、CNN.com 受到分散式阻絕服務(DDoS，Distributed denial of service)攻擊，造成各商業網站損失慘重，以及 2001 年的白宮網站被大陸駭客攻擊[9]，在在顯示出目前網路服務受到嚴重威脅，並且網路安全是非常薄弱的。

雖然由以上案例可知道目前駭客通常攻擊政府單位或是各大商業網站，但是駭客往往利

用個人主機本身的漏洞來發動攻擊。根據電腦網路危機處理暨協調中心(CERT/CC)網站所統計發現，作業系統本身漏洞數目連年成長(如圖 1.1)[3]，並且網路安全回報次數也逐年攀升(如圖 1.2)[4]。這些顯示出作業系統、應用程式具有許多漏洞，駭客透過這些個人主機上的漏洞而控制為跳板對象，進而發動分散式阻絕服務攻擊，導致癱瘓網路並且造成社會經濟蒙受巨大損失。

然而現今並無有效的防禦分散式阻絕服務攻擊，並且以目前的分散式入侵偵測系統而言，將多數的發生事件傳送給少部份之事件分析器，並統一在集中的分析系統中。以這些為分散偵測卻是集中分析的架構，根據國外的研究發現，目前已經無法應付現今具有隱藏性以及高度互助合作的分散式阻絕服務攻擊。此外目前根據國外的研究發現[23]，分散式入侵偵測系統已達到可擴充性的限制，如果攻擊者控制其中一個入侵偵測主機，則網路上的入侵偵測系統將被瓦解，攻擊者將可以在組織內無所忌憚找出弱點而不被發覺。

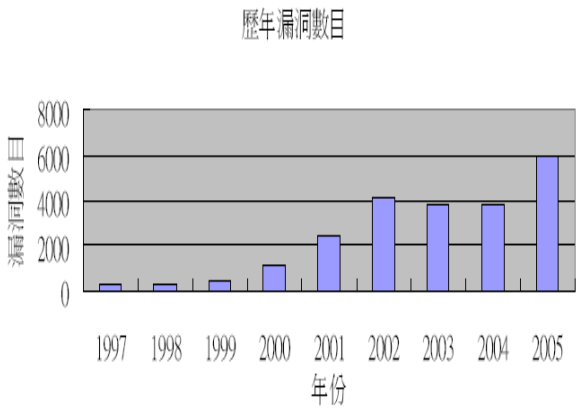

圖 1.1 1997~2005 年 CERT 歷年漏洞數目報告

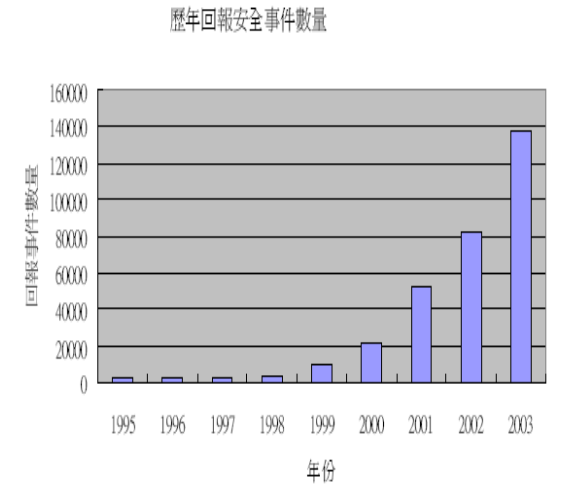

圖 1.2 1995~2003 年 CERT 所收到歷年安全事件數目

由於目前的洪水式分散阻絕服務攻擊尚無有效防禦機制，並且當入侵偵測系統被癱瘓時無法做出即時的備援。再者，根據過去的研究發現，由於現行的入侵偵測系統有以下缺點[12, 16, 21, 39]：

(1) 缺乏效率：入侵偵測系統需要及時反應發生事件，然而在現今的網路環境下，有大量的事件，造成入侵偵測系統無法及時反應。
(2) 誤報與誤判率過高：目前許多入侵偵測系統遍布在各企業的單一主機、應用程式或是單一的網路介面，以偵測並分析事件。然而錯誤的誤判率、攻擊行為都是較差的。
(3) 較高的維護成本：因為入侵偵測系統內部有規則資料庫，必須經常更新法則，使得誤判或是誤報率大幅降低，因此需要大量的維護成本。
(4) 有限的彈性：通常入侵偵測系統需在特殊的環境，或是為了改進瓶頸而修改內部偵測政策。然而必須每次將入侵偵測系統重新開始才能有效發揮，導致偵測系統缺乏彈性。

有鑑於以上目前入侵偵測系統的缺點，近年來有許多學者提出代理人應用在入侵偵測系統上。由於代理人具有蒐集資料以及與節點活動功能，因此符合新型分散式入侵偵測系統之特性，並且根據 Spafford 研究發現，將代理人應用在分散式入侵偵測系統上有以下優點[32, 39]：

(1) 降低網路延遲：由於代理人可透過繞徑方式使得網路延遲 大幅降低。
(2) 降低網路傳輸量：由於代理人只需針對必要資料蒐集，因此將可降低網路傳輸量。
(3) 高度擴充性：由於代理人本身特性，可針對網路環境而所調整，並且具備非同步與自動執行的能力。

是故，本研究基於多代理人以提出三種防禦機制，第一為基於橢圓曲線公開金鑰密碼系統驗證機制，以驗證合法使用者。其次為轉移服務埠機制，當大量封包傳送時將啟動轉移服務埠機制，期使服務正常運行。最後為備援機制，當主機被攻擊時，代理人將通知鄰近主機準備備援，經由此三種防護以真正達到防禦洪水式分散阻絕服務攻擊。

2. 文獻探討

2.1 多代理人

所謂多代理人(MA, Multi-agent)就是指一種具自動化的軟體程式，並且多個代理人具有自動蒐集資料，將所攜帶之資料回傳至主機上，進而分析資料內容。目前應用代理人系統可分為單一為單一代理人系統(Single-Agent System, SAS)及多重代理人系統(Multi-Agent System, MAS)。以下根據國外學者研究針對這兩種系統做進一步說明[34]：

(一)單一代理人系統：應用在此系統為單一代理人，本身由於是中央集權式的代理人系統，因此具有開發技術較為簡單、並無溝通上之困難。相對的，處理範圍有限、針對複雜度較高的工作無法執行，並且資料量繁雜時將無法有效處理。

(二)多代理人系統：此系統代理人數目為兩個以上，主要應用在分散式網路環境。多代理人系統是由數個具有(半)自主權並且交互作用的代理人所組成的分散式的電腦系統。多個代理人將可工作細分，將可整合異質資料來源，並且提供一致的操作環境。

2.2 阻絕服務攻擊

所謂阻絕服務攻擊(DoS Attack，Denial of Service Attack)通常是利用傳送大量資料封包，然而傳送地方都是偽造的位址。當遭受攻擊，並且超過網路負荷與最大連線時，對方主機與網路隨即癱瘓，如此一來遭受攻擊主機將無法正常提供服務。根據國外學者研究發現以下將阻絕服務攻擊手法分為五種[9, 12]，如圖 2.1。

(一)網路裝置層：針對網路上之設備，將內部應用程式漏洞加以攻擊以達到阻絕服務之能力。

(二)作業系統層：主要是利用作業系統特性來做攻擊，例如根據國外學者研究中發現，ICMP(Internet Control Message Protocol) 特性來攻擊[12, 18]，ICMP 是用來傳遞網路控制訊息的一種通訊協定。在 UNIX 環境有 ping 這個指令，主要傳輸回應封包，因此利用此特性將傳出大量回應封包，直至服務癱瘓為止。

(三)應用程式層：駭客利用惡意的應用程式移植至主機，使得往後受害主機做為跳板。

(四)大量資料攻擊：駭客利用網路頻寬大量傳輸資料，導致耗盡網路頻寬資源，進而達到攻擊手段。

(五)網路協定攻擊：駭客利用網路協定的漏洞來發動阻絕服務攻擊。

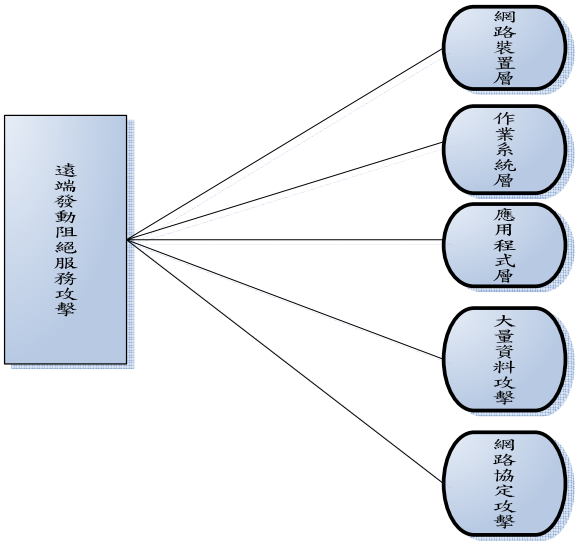

圖 2.1 遠端阻絕服務攻擊種類

2.2.1 分散式阻絕服務攻擊

阻絕服務攻擊由於是利用單點發送偽造的大量網路封包，使用簡單的流量監測系統就能根據流量找到攻擊者，為了避免這個缺點及更好的阻絕服務攻擊效果，根據國外的學者的研究發現，駭客們將原本單純的阻絕服務攻擊改成多層式架構(multi-tier)並發展廣佈型多點技術[12]，如圖 2.2。所以透過流量監測系統的統計過濾後，只能將駭客發動攻擊的區域縮小到較小的 IP 位址範圍，而且真的找出這些 IP 位址，大都也只是一個中繼站，真正幕後的駭客，可能遠在層層的中繼站後面，而對攻擊者而言由於封包位址是偽造的，所以要由 IP 位址來反追查(Trace back)更是困難重重。

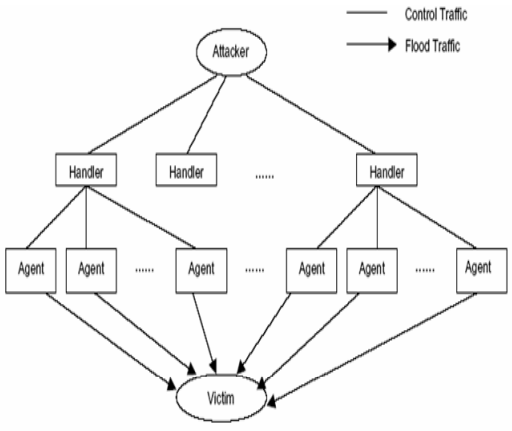

圖 2.2 DDoS 攻擊架構

分散式阻絕服務 DDoS(Distributed Denial of Service)攻擊，是透過主控端控制許多分散的已入侵主機，當收到攻擊指令時，一起對目標電腦發出大量的封包，目標電腦在流量過大的情形下，就會發生一般使用者無法得到服務，或是造成目標電腦當機而無法提供服務，由於是藉由分散的電腦來攻擊，所以有『分散式』阻絕服務的名稱。

2.2.2 攻擊步驟

根據國外學者的研究發現，攻擊者從準備到開始攻擊必須遵循四步驟[12]：

(一)選定代理者

首先攻擊者在全球資訊網上用掃瞄工具尋找一個或多個代理攻擊的主機(master)再由這些代理者控制更多的主機(slave)；代理者的位置，避免在受害主機的網域內(Target Domain)，是為了不讓受害主機(victim)能即時有效率的反應；而他們的位置不在攻擊者的網域內(Source Domain)，是為了逃避被追蹤發現攻擊者的位址，如圖 2.2；這些代理主機具有豐富的資源，例如：硬體性能強，連線頻寬大；初期用手動來控制，攻擊時已成為定時自動控制。

(二)滲透控制

根據國外的學者研究發現，攻擊者經由安全的漏洞，植入攻擊碼(如 Code Red)[21]；通常利用重新命名檔案、隱藏、變換位置等方法來躲避偵測，再以掃瞄工具(Scanning tools)探知有那些主機被屈服以及尚未被察覺，利用持續增加代理者的數量，來達到高破壞能力。

(三)建立聯繫通道

攻擊者在發動攻擊前，必須在攻擊者(attacker、client)、控制者(master、handler)、代理者(slave、daemon、agent)及受害者（victim）四者間建立安全的聯繫通道，這通道可以用加解密的方式來達到資料傳遞的隱密性。

(四)全面發動攻勢

攻擊者掌控所有的代理者後，設定固定的時間，同時間發動全面性的攻擊；攻擊時利用假造的來源位址(source address)、封包類型(packet type)、標頭區塊(header fields)來避免被偵測發現；並且不斷變換攻擊者、控制者與代理者間的聯繫通道，使其追蹤更加困難。

2.2.3 現今分散式阻絕服務攻擊方式

阻絕服務攻擊就是企圖阻止合法的使用者使用該項服務功能，分散式阻絕服務攻擊就是多台電腦在同一時間內對提供服務的主機進行阻絕服務攻擊。將分散式阻絕服務攻擊作了整體性的分類[12, 22]，如圖 2.3，以下將從散佈攻擊程式的手法和設計原理的角度去探討分散式阻絕服務攻擊的特性。

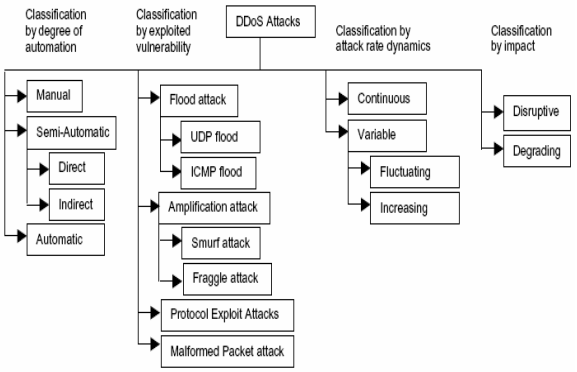

圖 2.3 DDoS 攻擊方法分類

攻擊者讓代理人機器感染攻擊碼的方式可以分為：手動(Mutual)、半自動(Semi-Automatic)、自動(Automatic)。早期的手動散佈攻擊碼的方式是攻擊者一一入侵存有漏洞(Vulnerability)的主機並且植入攻擊碼後在同一時間執行這些攻擊程式；半自動散佈攻擊碼的方式則是利用主從式(Master/Slave)架構，從操作者(指 Master)透過指令將攻擊碼散佈到代理(Slave)機器上，並且在同一時間對這些代理機器下達攻擊指令。

分散式阻絕服務攻擊大致上是根據現有的漏洞進行攻擊，大致上可以區分成兩類[8, 12]：

(一) 協定攻擊(Protocol Attack)

協定攻擊是利用安裝在主機上協定的特色或錯誤(Bug)造成大　量的使用主機資源；像 TCP SYN 攻擊是針對 TCP 在建立連線的過程中會產生三方交握(Three-Way Handshake)的問題攻擊者一直針對主機送出 SYN 封包卻不回應主機送來的訊息，直到儲存連線的佇列(Queue)被這樣的訊息佔滿。像 CGI 要求攻擊、伺服器認證攻擊也都是這一類的攻擊手法。

(二) 暴力攻擊(Brute-Force Attack)

暴力攻擊是利用大量合法的傳輸進行攻擊，因為在上游(Upstream)網域可以處理的網路流量到了受攻擊的網域會造成主機大量耗

盡資源。根據封包內容可以將暴力攻擊分為可過濾(Filterable)和不可過濾(Non-Filterable)攻擊：(a)可過濾攻擊：

可過濾攻擊是使用偽造的封包或無關主機運作緊要的封包，像 UDP 洪氾攻擊(UDP Flood Attack)、ICMP 要求洪泛攻擊(ICMP Request Flood Attack)可以被防火牆過濾。(b)不可過濾攻擊：不可過濾攻擊是使用合法的封包對伺服器提出服務要求，攻擊者和一般合法的使用者所送達大量的封包就造成阻絕服務攻擊。

2.2.4 洪水式分散阻絕服務攻擊

本研究主要防禦洪水式分散阻絕服務攻擊，因此將介紹 SYN Flooding Attack 特性。由於 SYN Flooding Attack 是針對 TCP Three-way handshake 漏洞而攻擊，因此以下先介紹 TCP Three-way handshake。

(一) TCP Three-way handshake[9]

當雙方主機要傳輸資料時，將先建立連線。而這個連線的過程是透過帶有控制旗標的封包在兩端主機間傳輸，也就是所謂的 Three-way handshake(圖 2.4)[1]，首先，起始端 S 先送一個帶有 SYN 控制旗標的封包到目的地 D，接下來，D 收到 S 送過來的封包之後，會回給 S 一個帶有 SYN 與 ACK 這兩個控制旗標的封包給 S，最後，起始端 S 收到 D 回給它的封包之後，再回給 D 一個帶有 ACK 控制旗標的封包，通知目的端。當 D 收到之後，兩端的連線就建立完成，並且可以開始傳輸資料。

Three-way handshake 也會初始化 S 和 D 兩端的 sequence number 作為可靠性傳輸之用，S 先送一個初始的 sequence number SYN_x 給 D，D 收到之後會回一個 D 初始的 SYN_y 加上回 S 的 ACK_{x+1} 給 S，S 收到 D 的 $SYN_y + ACK_{x+1}$ 後，會回給 D 一個 ACK_{y+1}。

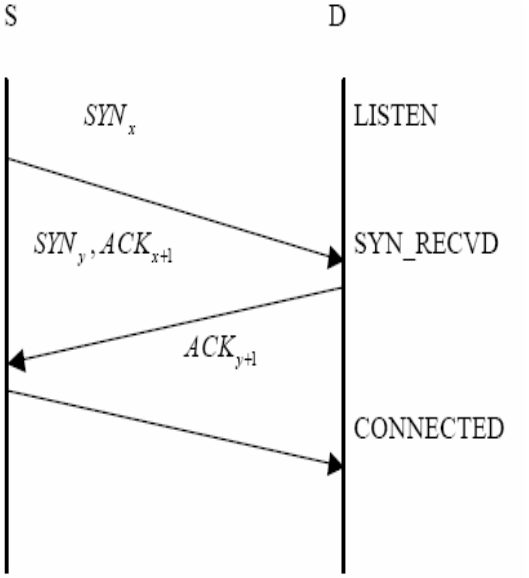

圖 2.4 TCP Three-way handshake

(二) SYN Flooding 阻絕服務攻擊

駭客對目標主機發送一連串 SYN 封包，每個封包都要求目標主機系統回應一個 SYN+ACK 封包(圖 2.5)[9, 11]，基本上這些封包的來源位址都是經過偽造的，接下來目標主機系統在回應 SYN+ACK 封包後會等待對方送出 ACK 封包並且會等待一個計時器。由於來源位址是偽造的，所以除非該來源位址有機器存在，不然不會有任何 ACK 封包送回給目標主機，因此目標主機的系統佇列裡面會暫存大量半連線狀態(half-open)連線，這連線請求必須等到收到對方的 ACK 封包或是超過逾時時間之後才會被移除。

當 Backlog 佇列達到上限之後，主機便會開始丟棄接下來任何的連線請求封包，讓正常的使用者沒有辦法存取該主機。要注意的是，如果在攻擊之前就已經建立的連線，此時仍然可以繼續存取該服務。此攻擊主要是因為 TCP 對於封包的管控上並沒有提供一個有效的認證機制，使得駭客可以很輕易的對這個弱點做攻擊。

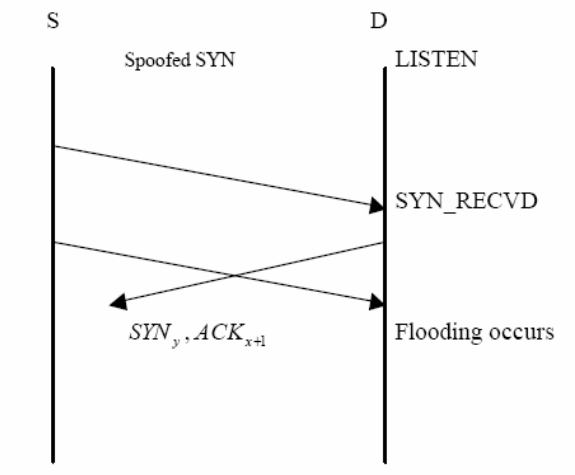

圖 2.5 SYN Flooding 阻絕服務攻擊

2.2.5 現行洪水式阻絕服服務攻擊防禦手法

直至現今已有許多抵擋 SYN Flooding Attack 的方法，包括主動式防禦[31]、修正防火牆設定[31]、修正路由器設定[12]、修正系統設定[9]、負載平衡[19]以抵抗此攻擊、改進連線機制[8, 12]，但仍有許多不足之處，以下將介紹上述防禦方法並說明尚存缺點：

(一) 主動式防禦

此方法主要是利用代理人方式，傳送代理人到各主機去蒐集封包資訊，並且分析封包而判斷是否為攻擊封包，然而此方法只能抵抗非法來源位址封包無法防禦合法使用者的攻擊封包。

(二) 修正防火牆設定

此方法主要是利用時間的延遲來做修正，若為非法的使用者則會回應 Rst 封包使得連線中斷，然而此方法針對合法的使用者但反應較久的話，合法的使用者將無法存取服務，並且當合法的使用者做攻擊時此方法無法防禦。

(三) 修正路由器設定

此方法主要是在路由器端口地方做過濾的動作，當非為此網域之使用者位址將阻擋封包進入，因此可以抵擋非法使用者偽造來源位址攻擊，然而此方法也是無法抵抗合法使用者攻擊。

(四) 修正系統設定

此方法將系統的 blacklog 容量加大或是縮減等待時間，使得可以提升系統處理封包能力與處理速率。然而此方法是消極的，因為將會耗費更多的建置成本，以及無法真正達到防禦之功能。

(五) 負載平衡

此方法主要精神為當偵測到偽造來源位址封包時，將此封包轉移至頻寬較小的伺服器上，正常封包則轉至頻寬較大的伺服器上，然而此方法將耗費更多的建置成本，以及無法真正達到防禦功能。

(六) 改進連線機制

此方法主要利用單向雜湊加密方式以驗證來源端之合法性，然而此方法雖能確實阻止惡意來源端攻擊，並且 MD5 已被大陸學者王小雲教授破解，因此在安全上有所缺失。

2.3 橢圓曲線密碼系統

橢圓曲線密碼系統是由 Koblitz and Miller 兩位學者所提出來的[20, 27]。在有限場 F_p 之下，給定橢圓曲線 E 上的兩個點 P 及 Q，當點 P 的序(order)夠大時，要找出一個整數 x，使得 $Q = x \cdot P$ 是很困難的，此問題稱為解橢圓曲線離散對數問題 (Elliptic Curve Discrete Logarithm Problem；ECDLP)。在 RSA 與 ElGamal 系統中需要使用 1024 位元的模數[13, 28]，才能達到足夠的安全等級，而 ECC 只需要使用 160 位元的模數即可[7, 38]。

ECC 的運算效率，相較於使用大指數運算的 RSA 機制而言，其所運算的時間少了許多。此外，在 RSA 系統中每一個使用者 U_i，其分別需要產生一組 $N_i = p_i \cdot q_i$，而在 ECC 中，橢圓曲線方程式及其生成數是可以重複使用，不像 RSA 對每一個使用者都必須個別儲存一份公開資訊。

表 2.1 列出分別以 ECDSA(Elliptic Curve Digital Signature Algorithm)[17]、ElGamal[13]、Schnorr[36]與 RSA[28]方式建立公開金鑰密碼系統的比較。由於橢圓曲線密碼系統所使用的金鑰長度相對較短，但是卻可以達到相同的安全等級[7, 38]。同時，因所需金鑰長度較短，故可提高加/解密的效能及減少儲存的成本。

表 2.1 各種公開金鑰密碼系統之比較

公開金鑰系統 / 比較項目	ECDSA	RSA	ElGamal	Schnorr
加/解密處理	使用不同金鑰	使用不同金鑰	使用不同金鑰	使用不同金鑰
公/私鑰可否互換	不可	可	不可	不可
明文攻擊防護機制	具備	欠缺	具備	具備
金鑰相對長度	短	長	長	長
加/解密理論	已公開	已公開	已公開	已公開
安全理論	解橢圓曲線離散對數問題	分解因數問題	解離散對數問題	解離散對數問題

2.4 小結

綜觀 2.2 節中前五種防禦方法都無法抵抗合法使用者攻擊，雖然第六種改進連線機制可以有效驗證來源端之合法性，將有效阻擋攻擊，但其作法為一般的單向雜湊加密方式，因此本研究將補其安全性，使得能偵測到當系統被攻擊至癱瘓時將無法正常提供服務。是故，本研究整合三種防禦機制，即基於橢圓曲線公開金鑰密碼系統驗證來源端位址、其次為轉移服務埠機制，抵抗當合法使用者攻擊時，服務仍然可以正常提供，最後為備援機制來備援偵測系統癱瘓時可以即時備援。相信結合以上三種防禦機制將可以真正達到防禦洪水式分散阻絕服務攻擊。

3. 研究方法

由於洪水式分散阻絕服務的強大破壞性，綜觀目前的防禦方法無法有效的解決此類攻擊，因此本研究以多代理人架構並且提出整合三種防禦機制，第一種為植基於橢圓曲線公開金鑰密碼系統之驗證機制、其二為轉移服務埠機制、最後為備援機制。當可疑封包進入時，過濾代理人將告知偵測代理人以啟動防禦機制，即使多代理人入侵偵測系統被攻擊，偵測代理人將啟動第三種防禦機制，告知兩側以備援癱瘓主機，如此三種防禦才能真正達到防禦洪水式分散阻絕服務攻擊。以下將在 3.1 中介紹本研究所提之多代理人架構，3.2 中詳述本研究之多代理人架構之洪水式分散阻絕服務攻擊偵防機制，以及在 3.3 中說明本研究所提之三種防禦機制。

3.1 多代理人架構

由於分散式阻絕服務攻擊需要有效偵測、有效過濾封包，並且當主機受到攻擊時能有效備援。是故，本研究基於多代理人架構[35]以達到偵測、過濾、備援之能力，並且發揮有效分工。

其多代理人架構中，第一個是過濾代理人，主要工作為監控封包異常與過濾封包，並且將異常紀錄更新至知識庫中，最後將訊息傳送給偵測代理人。第二個為偵測代理人，主要為巡邏偵測主機是否異常，若發現主機異常將啟動防禦機制，並通知備援代理人準備備援工作，以及將過濾代理人所傳送主機訊息給下一個代理人平台。如圖 3.1 中，平時偵測代理人將會依序移動至各代理人平台，若有異常狀況將會告知備援代理人。第三個為備援代理人，當主機發生異常時，偵測代理人將通知兩旁備援代理人，選擇主機資源負荷較小的為主要備援主機，使得當被攻擊至癱瘓時可以即時備

援，而不因癱瘓導致服務終止。圖 3.2 為多代理人架構。

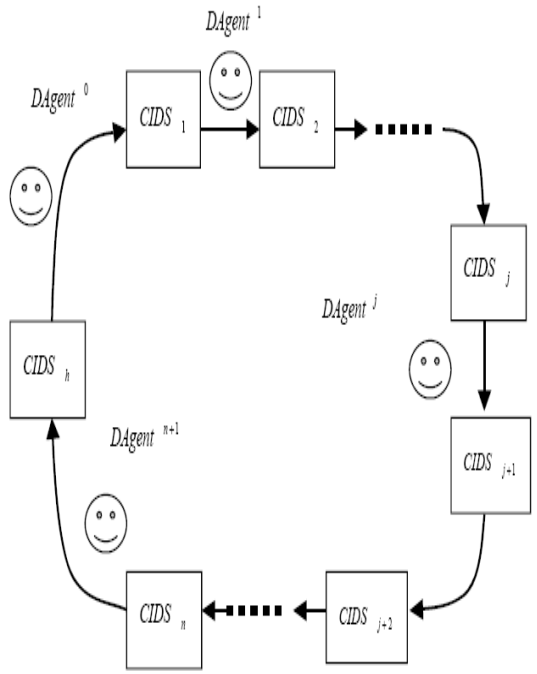

圖 3.1 偵測主機送出代理人給各被偵測主機的路徑

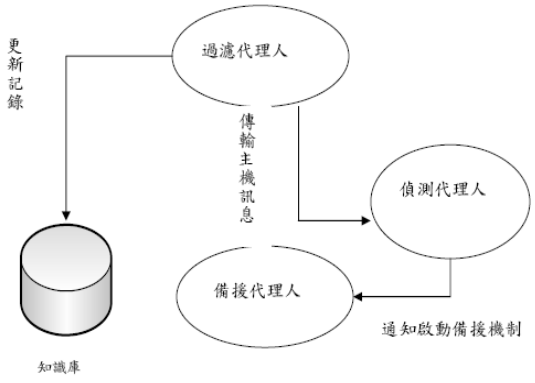

圖 3.2 多代理人架構運用在入侵偵測之架構

3.2 多代理人架構之洪水式分散阻絕服務攻擊偵防機制

以下將介紹本研究所提出之基於多代理人架構之洪水式分散阻絕服務攻擊偵防機制：

(一)在圖 3.3 中，當外部、內部封包流入時，過濾代理人將啟動，若流量超過所設定的門檻值，將啟動防禦機制，並且過濾代理人將紀錄位址。

(二)當過濾代理人發現這些不正常流量進入網域時，將阻止瞬間突然暴增的封包。

(三)過濾代理人將異常狀況以及紀錄攻擊位址更新至知識庫中。

(四)過濾代理人將主機訊息傳送給偵測代理人，由偵測代理人攜帶訊息至下一個代理人平台。

(五)偵測代理人將所帶回來的紀錄更新至知識庫中。

(六)若發生異常，偵測代理人將與備援代理人溝通，並且啟動備援機制。

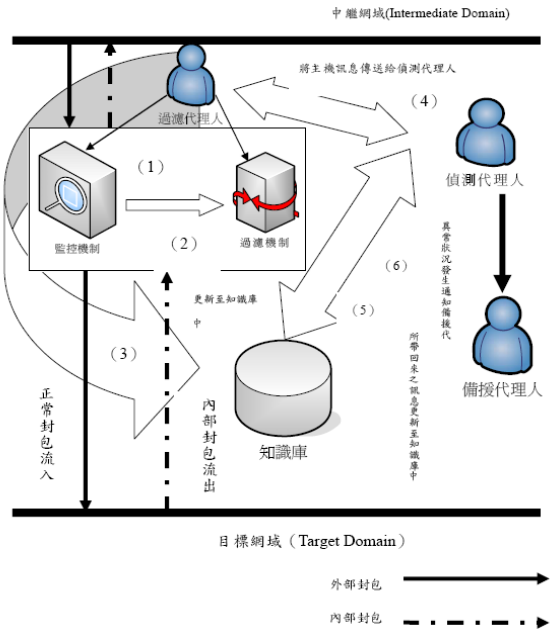

圖 3.3 多代理人入侵偵測系統

當外部封包流入本系統時，過濾代理人將執行過濾機制，並且執行防禦機制，本研究整合三種防護以達到完整防禦機制。第一種防護以橢圓曲線公開金鑰密碼系統做為驗證機制來防禦非法主機，第二種防禦當偵測為不正常封包時，偵測代理人將告知主機關閉服務埠，並隨機式開啟新的服務埠並以 ECC 加密告知新的服務埠，第三種防護當入侵偵測系統被攻擊至癱瘓時，偵測代理人將告知兩邊主機啟動備源機制，即使入侵偵測系統被攻擊也將正常執行防禦機制。以下將依序介紹本研究所提的三種防禦機制。

3.3 防禦洪水式分散阻絕服務攻擊機制設計

當外部封包流入入侵偵測系統時將啟動過濾機制，其說明如下(圖 3.4)：

步驟 1. 外部封包進入入侵偵測系統。

步驟 2. 封包是否為 SYN+ACK 封包，以及封包量是否大於門檻值

步驟 3. 若步驟 2 成立，則啟動 ECC Scheme 以達到驗證功能，其中 ECC Scheme 將在 3.3 節中介紹。

步驟 4. 若步驟 2 不成立，但封包量大於門檻值則執行第二種防禦機制，以上皆不成立則放行封包。其中第二種防禦機制將在 3.3.2 節說明其做法。

步驟 5. 執行 ECC Scheme 後，若為非法使用者則將對方 IP 紀錄至知識庫中。

以下將介紹此過濾機制之演算法，首先介紹此演算法之參數如下：

Algorithm Notation

packets_income()：流入外部封包
packets_income(type)：封包型態
SYN_ACK：封包型態為 SYN+ACK
packets_income(num)：流入封包大小
ECC_Scheme()：橢圓曲線公開金鑰密碼機制
packets_income(address)：流入封包之來源端位址
TransPort()：轉移服務埠機制
port_num：服務埠數值
pass(packets_income)：放行外來封包
adddb()：更新知識庫

Algorithm：*Filtering the packets*

Input：IncomingPackets　　Ouput：NormalIncomingPackets

```
1   while(packets )
2    if((packets_income(type)= =SYN_ACK))&&(packets_income(num)>=50))
3      ECC_Scheme(packets_income(address))
4        else if((packets_income(num)>=50))
5            TransPort(port_num)
6        endif
7   else
8    pass(packets_income)
9   endif
10   if(ECC_Scheme(packets_income(address)= =FALSE))
11      adddb(packets_income(address)
12   endif
13  endwhile
14  End
```

圖 3.4 過濾機制演算法

3.3.1 植基於橢圓曲線公開金鑰密碼系統防禦機制

當過濾為可疑封包時將執行橢圓曲線公開金鑰密碼系統機制以確認是否為合法客戶端，該防禦機制主要是利用 SYN Cookie 對 TCP 服務器端的三次握手協定作一修改。當收到 TCP SYN 封包時並傳回 SYN+ACK 封包，根據 SYN 封包計算出一個 Cookie 值，在收到 TCP ACK 封包時，服務主機將根據 Cookie 值檢查 TCP ACK 的合法性。因此 Cookie 值不但帶有連接的屬性，並且是不可偽造的。以下將基於曹學者所提機制介紹此流程[37]：

(一)系統建置階段

系統所使用之參數如下：

1. p：有限場的大小，而其長度為 160 bits 的大質數。
2. 位於 $\mathbf{F}_p$ 上的橢圓曲線 E：

橢圓曲線方程式 $y^2=x^3+ax+b$，其中 $a, b \in F_p$ 且滿足 $4a^3+27b^2 \neq 0 \pmod p$。另外，其所有的點 (x,y)位於橢圓曲線 E 而形成 $E(F_p)$的集合，且包含一無窮遠點 O，其中 $x, y \in F_p$。

3. B：即序為 n 之橢圓曲線上的生成點，而 n 為長度 160 bits 的大質數。
4. P_i：使用者 U_i 的公鑰。
5. S_i：使用者 U_i 的私鑰。
6. P_{SA}：系統中心的公鑰，其中 $P_{SA}=S_{SA} \cdot B \pmod p$ (”・”意味著一個數值乘上一個橢圓曲線上的點)。
7. S_{SA}：系統中心的私鑰，其中 $S_{SA} \in [2, n-2]$。
8. SA：系統中心，此系統為每個區域 CIDS。
9. $h()$：單向雜湊函數 (One-way hash function)，其輸出為一固定長度值 j，$j \in [2, n-2]$，而其長度為 160 bits。此單向雜湊函數必須滿足的單向雜湊函數之特性：若 $x' \neq x$，則不可能計算出 $h(x') = h(x)$，亦即 $h(x') = h(x)$ 若且為若 $x' \neq x$。
10. w_i：使用者 U_i 的公鑰證明(Witness)。

(二)使用者註冊階段

使用者向系統中心 SA 註冊以取得公鑰與私鑰，其過程如下：

1. 使用者 U_i 隨機選擇一個亂數

$x_i \in [2, n-2]$ 作為主鑰，且計算 $V_i = h(x_i \| I_i) \cdot B$，並將自己的身分 I_i 與 V_i 傳送給 SA。

2. SA 隨機選擇一個時間變數 $k_i \in [2, n-2]$，且計算使用者 U_i 的公鑰 $P_i = V_i + (k_i - h(I_i)) \cdot B = (P_{ix}, P_{iy})$ 與公鑰證明

$w_i = k_i + S_{SA} \cdot (X(Pi) + h(Ii)) (mod\ n)$，並回傳 P_i、w_i 給使用者 U_i。

3. 使用者 U_i 計算自己的私鑰 $s_i = w_i + h(x_i || I_i) (mod\ n)$，並驗證公鑰的有效性是否成立：

$S_i \cdot B = P_i + h(I_i) \cdot B + [(X(P_i) + h(I_i))\ mod\ n] \cdot P_{SA}$ (1)

若驗證成功，則使用者 U_i 的私鑰為 S_i，公鑰為 P_i。

定理 1：使用者於註冊階段所得到的來自於 *SA* 之公鑰 P_i 可由方程式(1)驗證其有效性。

證明:

$$\begin{aligned} s_i \cdot B &= (w_i + h(x_i \| I_i)) \cdot B \\ &= w_i \cdot B + h(x_i \| I_i) \cdot B \\ &= (k_i + s_{SA} \cdot (X(P_i) + h(I_i))) \cdot B + h(x_i \| I_i) \cdot B \\ &= (k_i + h(x_i \| I_i)) \cdot B + (X(P_i) + h(I_i)) \cdot P_{SA} \\ &= k_i \cdot B + V_i + (X(P_i) + h(I_i)) \cdot P_{SA} \\ &= P_i + h(I_i) \cdot B + [(X(P_i) + h(I_i)) \bmod n] \cdot P_{SA} \end{aligned}$$

□

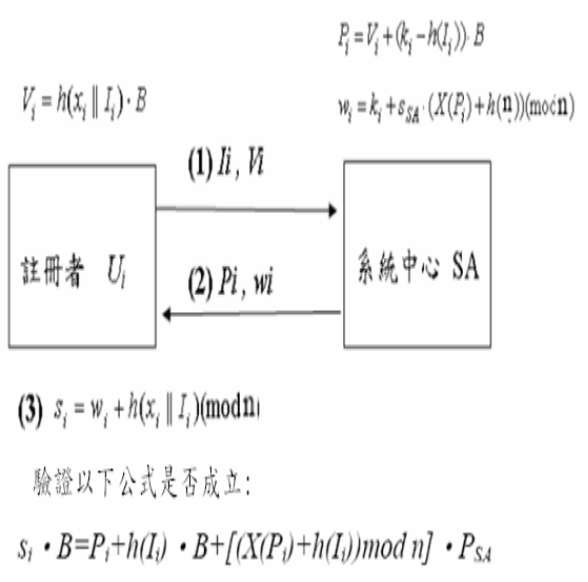

圖 3.5 使用者註冊階段

(三)基於 SYN Cookie 之 ECC-based TCP 三方交握協定

圖 3.6 為基於 SYN Cookie 之 ECC TCP 三方交握協定，以下將介紹客戶端 U_j 傳送 SYN 封包給服務主機 U_i，其中 seq 為序列號值，並且以 ECC 來達成驗證合法客戶端，其步驟如下所示：

步驟 1：客戶端 U_j 任選一個隨機整數 $w \in [2, n-2]$，並加密其 SYN 封包 S，以及計算 V_i、C_1 與 C_2 如下：

$$V_i = P_i + h(I_i) \cdot B + [(X(P_i) + h(I_i)) \bmod n] \cdot P_{SA}$$

$$C_1 = w \cdot B,$$

$$C_2 = S + w \cdot V_i$$

U_j 將 C_1 與 C_2 傳送給解密者 U_i

步驟 2：U_i 取得 C_1 與 C_2

U_i 計算

$$\begin{aligned} &C_2 - s_i \cdot C_1 \\ &= S + w \cdot V_i - (w \cdot s_i \ \ mod\ n) \ \cdot B \\ &= S + (w \cdot s_i \ \ mod\ n) \ \cdot B - (w \cdot s_i \ \ mod\ n) \ \cdot B \\ &= S. \end{aligned}$$

步驟 3：服務主機產生 Cookie 後，隨機選取隨機整數 w_2 並將加密 SYN+ACK 封包傳送給客戶端 U_j。以及計算 V_j、C_3、C_4 如下：

$V_j = P_j + h(I_j) \cdot B + [(X(P_j) + h(I_j))\ mod\ n] \cdot P_{SA}$

$$C_3 = w_2 \cdot B$$

$$C_4 = SYN\_ACK + w_2 \cdot V_j$$

U_i 將 C_1 與 C_2 傳送給解密者 U_j

步驟 4：U_j 取得 C_3 與 C_4

U_j 計算

$$\begin{aligned} &C_4 \ - s_j \cdot C_3 \\ &= SYN\_ACK + w_2 \cdot V_j - (w_2 \cdot s_j \ \ mod\ n) \ \cdot B \\ &= SYN\_ACK + (\ w_2 \cdot s_j \ \ mod\ n) \cdot B - (w_2 \cdot s_j \\ &mod\ n) \ \cdot B \\ &= SYN\_ACK\ . \end{aligned}$$

步驟 5：客戶端 U_j 任選一個隨機整數 w_3，並加密其 ACK 封包，以及計算 V_i、C_5 與 C_6 如下：

$$V_i = P_i + h(I_i) \cdot B + [(X(P_i) + h(I_i)) \bmod n] \cdot P_{SA}$$

$$C_5 = w_3 \cdot B,$$

$C_6 = ACK + w_3 \cdot V_i$

U_j 將 C_5 與 C_6 傳送給解密者 U_i

步驟 6：U_i 取得 C_5 與 C_6

U_i 計算

$C_6 - s_i \cdot C_5$

$= ACK + w_3 \cdot V_i - (w_3 \cdot s_i \mod n) \cdot B$

$= ACK + (w_3 \cdot s_i \mod n) \cdot B - (w_3 \cdot s_i \mod n) \cdot B$

$= ACK$

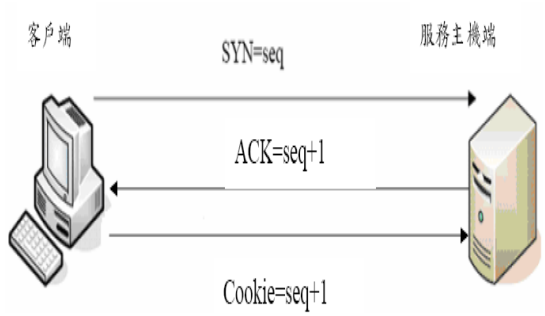

圖 3.6 基於 SYN Cookie 之 ECC TCP 三方交握協定

3.3.2 轉移服務埠防禦機制

當流入封包型態為其他型態，並且為大量封包時，仍然會使服務中斷而影響主機運行，因此本研究提出第二種防禦為自動轉移服務埠機制，其說明如下：

步驟 1. 外部封包進入。

步驟 2. 若封包量大於門檻值偵測代理人將傳送服務主機警告訊息，未大於門檻值則放行封包。

步驟 3. 服務主機立即關閉服務埠。

步驟 4. 服務主機隨機挑選新的服務埠。

步驟 5. 服務主機開啟新的服務埠。

步驟 6. 服務主機傳送加密過的服務埠訊息給客戶端。

步驟 7. 客戶端若為合法 IP 則將可解密取得新的服務埠。

以下將介紹此轉移服務埠防禦機制之演算法(圖 3.7)，其演算法參數說明如下：

Algorithm Notation

packets_income()：流入外部封包
packets_income(num)：流入封包大小
Sendagent()：偵測代理人傳輸訊息
Sendagent(alarm)：偵測代理人傳輸警告訊息
ServerReceive：服務主機接收訊息
ServerEncrpt()：服務主機執行加密程序
Newport：新的服務埠值
random()：隨機亂數
ECC_Scheme_Encrpt()：橢圓曲線公開金鑰加密機制
ServerSend()：服務主機傳送函數
ClientReceive：客戶端接收訊息
pass(packets_income)：放行外來封包
ClientDecrpt()：客戶端執行解密函數

Algorithm：*TransferPort*

Input：IncomingPackets　Ouput：GettingNewPort

```
1  while(packets_income )
2   if((packets_income(num)>=50))
3     ServerReceive=Sendagent(alarm)
4     ServerClose(port)
5     ServerEncrpt((Newport= =random()))
6     ServerStart(Newport)
7     ClientReceive =ServerSend(ECC_Scheme_Encrpt((Newport= =random())))
8   else
9    pass(packets_income())
10  endif
11 endwhile
12 ClientDecrpt(ClientReceive)
13 End
```

圖 3.7 轉移服務埠防禦機制演算法

3.3.3 備援機制

本研究第三種防禦方法為曹與吳兩位學者所提之備援機制[2]，當平時巡邏偵測代理人偵測到聯防式入侵偵測系統 CIDS(Cooperative Intrusion Detection System)被攻擊時，將通知兩側主機，並選擇負荷量最小為備援主機，即使其中主機被攻擊至癱瘓也能順利執行防禦機制。此備援機制將分為三部份，行動代理人偵測攻擊、行動代理人偵測流程以及備援方法，將詳述如下：

(一)行動代理人偵測攻擊方法

現行有許多行動代理系統如 Aglets、Agent TCL 等，其使用參數介紹如下：

1. bc：agent 執行的二進位碼。

2. Md^j：agent 在 $CIDS_j$ 主機執行任務後累計所有的資料（$Md^{j-1} \subset Md^j$），送回 $CIDS_h$ 始位主機（Home）參考，待會我們利用它來判定網路流量狀態是否正常以及是否遭受到攻擊。

$$Md^j = \left(md^j, fd^j, td^j\right) \quad (3.1)$$

(1) md^j：在 $CIDS_j$ 上的監控值，超過設定門檻值的為正，低於門檻值為負。

(2) fd^j：在 $CIDS_j$ 上的過濾值，當過濾器（Ingress Filtering）阻止突然暴增封包超過設定門檻值的為正，低於門檻值為負。

(3) td^j：在 $CIDS_j$ 上過濾器(Ingress Filtering)發現的惡意攻擊來源 IP 位址。

3. r：描述 agent 旅行途徑，包含所有必須造訪的 $CIDS$ 主機。

$$r= \left(cids_1,...,cids_j,...cids_n\right) \quad (3.2)$$

4. uid^j：每 $CIDS_j$ 主機產生的獨特回覆確認碼，當 $agent^j$ 傳送到下個 $CIDS_{j+1}$ 主機時回傳 uid^j 給 $CIDS_j$ 主機來確認是否正確到達。

5. $vc^{\#(cids_j)}$：描述已經造訪過的主機序列，$\#\left(cids_j\right)$ 為目前為止造訪過的主機數量。代理人 Agent 在始位 $CIDS_h$ 主機（Home）開始遷移前，$vc^{\#(cids_h)} = vc^0$ 主機序列是空的，當它開始造訪旅程時，在釋放 agent 前，經過 $CIDS_1$ 主機產生 $vc^{\#(cids_1)} = vc^1 = cids_1$ 序列。

6. rid_j：當 $CIDS_j$ 主機被攻擊經過重新啟動功能後，送給鄰近備援主機 $CIDS_{j-1}$ 與 $CIDS_{j+1}$ 的恢復確認碼，讓備援主機知道 $CIDS_j$ 已經恢復，結束備援功能。

7. voc：我們在一個代理人的旅程中可以指定獨立及非獨立兩種計算，在獨立計算裏不需要其他主機的資料來加以計算；相反地，在非獨立計算當中需要其他主機的資料輸入才可計算，如此可用限制造訪主機命令元件(visiting host constraints)*voc* 來規範。$voc = \left(voc_1, voc_2, ..., voc_n\right)$，$n$ 是在 r 造訪的 $CIDS$ 主機總數量。如果 $voc_i = 0$ 在 $cids_i$ 的計算就不需要其他的主機資料加以計算，例如：

$$r= \left(cids_1, cids_2, cids_3, cids_4, cids_5\right)$$

$$voc = \left(0,0,\{cids_1, cids_2\},0,\{cids_3, cids_4\}\right)$$

在 $CIDS_1, CIDS_2, CIDS_4$ 等造訪主機不需要其他的主機資料加以計算是屬於獨立計算，而在 $CIDS_3$ 主機中，他必須依靠 $CIDS_1, CIDS_2$ 主機的輸入值加以計算，所以 $CIDS_3, CIDS_5$ 是屬於非獨立計算。

8. m：限制無法接收 agent 的 CIDS 主機的最大容許數量，在選擇下一個主機的演算法(*SelectNextHost*)中使用。

9. 偵測代理人：在 $CIDS_j$ 主機上執行偵測後的狀態，包含代理人的資料數據。

$$DAgent^j = \left(bc, Md^j, voc, uid, r, vc^{\#(cids_j)}\right) \quad (3.3)$$

10.備援代理人：在 $CIDS_j$ 主機上執行備援後的狀態，包含代理人的資料數據。

$$BAgent^j = \left(bc, Md^j, rid_j, voc\right) \quad (3.4)$$

本研究利用行動代理人巡邏偵察、搜索及探測的方式來確認主機是否被攻擊，以及傳送假造的來源位址給受害者網域的 MAIDS(Multi-agent Intrusion Detection System)，達到知識分享的功能，首先從 $MAIDS_h$ 始位主機(*Home host*)送出服務($DAgent_j$)給各被探測的主機，按設定規劃的路線 r=(maids_1,...., $maids_j$,...., $maids_n$)行走，最終會回到 $MAIDS_h$ 始位主機如圖 3.1，在送到每一主機的代理元件($DAgent_j$)會隨著不同的主機而改變。如果經過(受測)主機內的防禦狀態超過設定門檻值，就拒絕偵測代理人的服務，由此判定主機已被攻擊或成為攻擊者。

(二) 代理人偵測流程

1.在圖 3.8 中，我們假設 $CIDS_3$ 主機遭受攻擊，當偵測代理人依序探測到 $CIDS_3$ 主機時，$CIDS_3$ 主機在執行接收協定時發現自己的

$\tilde{M}d^3$ 值超過設定門檻值，無法回覆確認碼 $Sig(uid_3)$ 給 $CIDS_2$ 主機。

2. $CIDS_2$ 主機立即再次執行選擇下一主機的演算法找出下個傳送的主機是 $CIDS_4$ 主機，如果 $CIDS_4$ 主機也被攻擊，$CIDS_2$ 主機再跳過 $CIDS_4$ 主機，找出下個傳送的主機是 $CIDS_5$ 主機，以此類推，本研究只探討到 $CIDS_3$ 主機被攻擊。

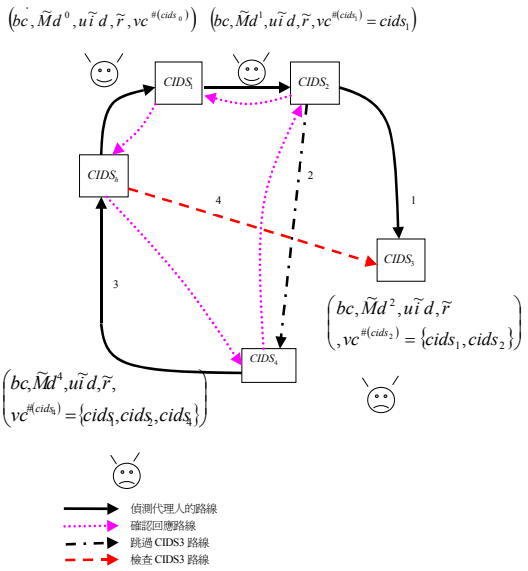

圖 3.8 偵測到 $CIDS_3$ 主機遭受攻擊

3.假設 $CIDS_2$ 主機截獲到攻擊 $CIDS_4$ 主機的假造 IP 位址，$CIDS_4$ 接收後就立刻存入知識庫內（knowledge base）讓 $CIDS_4$ 知道攻擊者的位址，並回覆確認碼 $Sig(uid_4)$ 給 $CIDS_2$ 主機，因為是跳過 $CIDS_3$ 主機，所以設定可以通過確認，$DAgent^4$ 內的造訪過主機集合的記錄資料為 $vc^{\#(cids_4)}=vc^3=\{cids_1,cids_2,cids_4\}$，代理人的資料會循序的送到下一主機，一直到 $CIDS_h$ 主機結束。

4.最後 $CIDS_h$ 主機將送回的資料加以判讀 $DAgent^h=\left(bc,\tilde{M}d^h,u\tilde{i}d,\tilde{r},vc^{\#(cids_h)}=\{cids_1,cids_2,cids_4,cids_h\}\right)$，發現 $CIDS_3$ 主機未在造訪主機的集合中，傳送一檢查程式給 $CIDS_3$ 主機再次確認已被攻擊。

(三)備援方法

當某一個 CIDS 主機內的監控值已超出設定門檻值，過濾器(Ingress Filtering)無法阻擋惡意封包時，我們利用備援的機制設定 $CIDS_j$ 主機的備援代理主機為 $CIDS_{j-1}$ 與 $CIDS_{j+1}$ 主機，每一個主機都以相鄰的兩個主機為代理主機，當 $CIDS_j$ 被攻擊不能完成偵防的任務時，相鄰的備援代理主機互相比較偵防狀態以選出合適的備援代理主機，來替補 $CIDS_j$ 被攻擊主機的任務。

1.傳送備援代理人演算法

Step1 Sent BAgent BackupHost found by Algorithm SelectBackupHost

Step 2 Until (no confirmation received and time-out)

wait for confirmation

Step 3 if (confirmation received and valid)

Step 4 store confirmation in local database

Step 5 else

go to step1

end

Algorithm: SelectBackupHost (at $cids_j$)

1 If in($cids_{j-1}$, $vc^{\#(cids_j)}$)

2 application terminate

3 else

4 If ($Md^j >= Md^{j-2}$) //判斷兩主機那個狀態較好

5 BackupHost = $cids_j$

6 else

7 BackupHost = $cids_{j-2}$

8 endif

9 endif

2.備援代理人接收確認碼演算法

1 Receive BAgent

2 If receive a rid_j //如果收到恢復碼立即

結束備援工作

Terminating Backup application

3 else

Backup //執行備援工作

Create and Send confirmation //送確認訊息

4 end

首先在傳送備援代理人中的第一行，以選擇備援主機(*SelectBackupHost*)的演算法決定由誰來備援，並且送出備援代理人($BAgent^j$)給備援主機；在選擇備援主機演算法中第一行，判別上個$CIDS_{j-1}$主機是否在旅程中來確認攻擊；第四行，比較自己$CIDS_j$與另一個備援代理主機$CIDS_{j-2}$主機的偵防狀態；在備援代理人接收確認碼的第二行，如果接收主機收到恢復碼rid_j，就立即停止備援工作，否則執行備援工作並送出確認訊息。

4. 安全性分析與探討

本研究基於多代理人以提出三種防禦機制，第一為基於橢圓曲線公開金鑰密碼系統驗證機制，以驗證合法使用者。其次為轉移服務埠機制，當大量封包傳送時將啟動轉移服務埠機制，期使服務正常運行。最後為備援機制，當主機被攻擊時，代理人將通知鄰近主機準備備援，經由此三種防護以真正達到防禦洪水式分散阻絕服務攻擊。本章將在 4.1 中分析所提機制之安全性，其 4.2 中將與各文獻所提之機制作逐一比較。

4.1 安全性分析

由於本研究所提機制中，前兩種是基於橢圓曲線公開金鑰密碼系統來加以保護，因此將在以下作安全性分析。

(1) 攻擊者無法取得使用者的主鑰 xi

使用者的主鑰 xi 在註冊階段中是透過 ZKP 所保護地，而其所基於的安全度，乃是 ECDL 與 OWHF 的困難度。即使在最糟的情況下，由於一些不可預料的人為因素，造成使用者的私鑰 si 被洩漏。例如，意外地在未認證的使用者前出示此私鑰，然而，即使這種情形發生，使用者的私鑰仍將被 OWHF 的困難度所保護。

(2) 攻擊者無法取得使用者的私鑰 si

攻擊者無法經由 si=wi+h(xi||Ii)(mod n)來推得使用者的私鑰 si，除非攻擊者能破解使用者在註冊階段所生的 xi，但其安全度是基於上述之 OWHF 與 ECDL 困難度。

(3) 攻擊者無法取得系統中心的私鑰 sSA

系統中心的私鑰 sSA 與相對應的公鑰 P_{SA} 是被定義在系統建置階段，而此一對金鑰只會用於註冊階段中產生使用者公鑰。若攻擊者要從 P_{SA} 中直接計算出 sSA，而這是很明顯地必須要面對 ECDL 困難度。另外，在使用者註冊階段中，sSA 是被時間變數 ki 所保護，而其安全性是基於 ECDL 困難度。因此，在此困難度下，系統中心私鑰 sSA 將無法透過公開資訊所計算出來。

(4) 達到安全等級 Level 3

系統中心有意假造 P_i'與 s_i'，但這會造成相同的使用者擁有兩個公鑰 P_i'與 P_i，此系統中心的詐騙行為會被偵測出來，因為本研究採用自我驗證公開金鑰密碼系統，使用偽造的 P_i'來進行的數位簽章/驗證簽章、加/解密或簽密法時並不會成功。

(5) 安全的加/解密機制

攻擊者要從加密者所傳遞的 C_1 與 C_2 解出明文，皆會遇到解橢圓曲線離散對數問題的困難度，另外加密者每次使用不同的隨機整數 w，因此攻擊者亦很難經由累積多個 C_1 與 C_2 而計算出明文。

4.2 本研究機制與文獻比較分析

(1) 惡意封包攻擊：由於本機制透過第一種驗證使用者防禦機制，可以驗證使用端傳送封包之合法性，是故，可抵擋惡意封包攻擊。

(2) 三方交握漏洞攻擊：本研究透過第一種驗證使用者合法性，可以驗證使用端之身分合法性。因此可防禦該漏洞攻擊。

(3) 合法使用者攻擊：本研究透過第一種驗證使用者防禦機制若被偽冒合法使用者攻擊，而使系統持續被攻擊時，將啟動第二種轉移服務埠防禦機制，將可抵擋合法使用者攻擊。

(4) 備援機制：當系統攻擊至癱瘓時，將即時啟動第三種備援機制，將可有效確保防禦系統之運行。

由以上可知本研究將可抵擋以上三種攻擊以及達到機動備援的功能，並且與各文獻比較結果如表 4.1 所示：

表 4.1 機制比較

	惡意封包攻擊	三方交握漏洞攻擊	合法使用者攻擊	備援機制
Schulba, Krsul, & Kuhn, 1997 [31]	○	X	X	X
Schulba et al., 1997 [31]	○	○	X	X
Douligeris et al., 2004[12]	○	X	X	X
Chen et al., 2006[9]	X	○	X	X
Kargl et al., 2001[19]	○	X	X	X
Chen, 2004[8]	○	○	X	X
本研究	○	○	○	○

○：可抵擋所提之攻擊，X：無法抵抗所提之攻擊

5. 系統實作與成果

本研究將在 5.1 中介紹本系統所使用之模擬測試工具，在 5.2 中介紹本實作所需之軟硬體規格，5.3 中說明所開發系統之操作流程，最後在 5.4 中詳述所提機制之安全性分析。

5.1 模擬測試工具

將以 NS2(Network Simulator - Version 2)模擬工具實作本研究機制，此工具為美國柏克萊大學網路安全實驗室研發而成[26]。圖 4.1 為 NS2 架構，圖中 OTcl 為主要用來撰寫網路拓撲部分的 script language，而 OTcl 其實就是架構在 Tcl 之上的 Object-Oriented 後所延伸出來的。在 Event Scheduler 和 Network Component 這二部分，主要是以 C++所寫出來的模組。NS 是以事件驅動的概念在做模擬。例如：在 1.2 秒的時候開始做 FTP 資料的傳輸，在 2.0 秒的時候結束傳輸[26]，亦即 Event Scheduler 的部分。至於 Network Component 則是 NS 中的 Agent (TCP、UDP...)、Traffic Generator (FTP、CBR...)。C++在最底層則是表示 NS 的核心程式語法。最後 Tclcl 為溝通 OTcl 及 C++的橋樑。

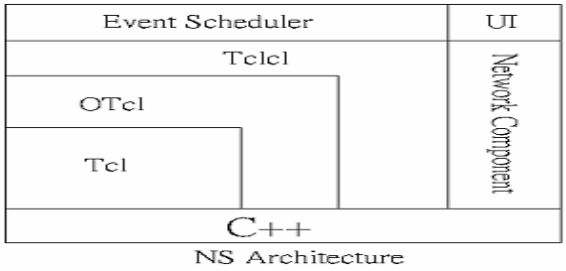

圖 5.1 NS2 架構

5.2 軟硬體規格

本機制所開發之系統所需軟體規格如表，而硬體規格如表所示：

表 5.1 軟體規格

軟體名稱	版本
Microsoft Visual Studio C++	Version 6.0
Network Simulator	Version 2
PSPad	Version 4.3.2
Windows XP	SP2

表 5.2 硬體規格

硬體名稱	規格
CPU	Pentium-D-830 3.0G
RAM	512MB
HD	200GB7200 轉/8M / S-ATA 介面
Network	10/100/1000 Base-TX Ethernet LAN

5.3 系統操作流程

由於此模擬工具是以 Linux 平台上之

Xwindow 介面呈現，所以必須在編譯畫面下達指令，編譯好之程式才會呈現出圖形介面。以下將介紹本研究所提之機制，其操作流程如下所示：

(1) 轉移服務埠機制

當確認為合法使用者時，但仍然傳送大於門檻值為 50 個封包大小時，主機將會自動傳送新的服務埠號碼給客戶端告知服務將由新的服務埠傳送，圖 4.2 中可知服務主機端會產生隨機亂數以及加密過之新服務埠數值，客戶端收到主機端傳送過來之訊息，並加以解密，所得到之數值為新的服務埠數值。其操作圖如下所示：

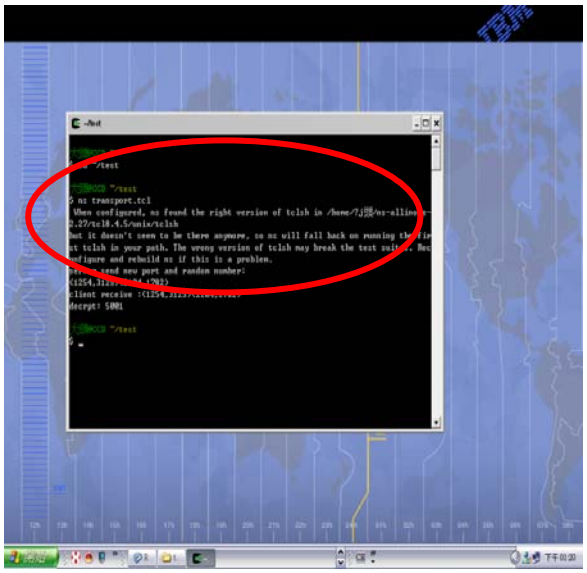

圖 5.2 轉移服務埠機制

圖 5.2 中首先在 NS2 的操作命令視窗 Cygwin 下達執行加密檔案指令 ns transport.tcl，程式將重新編譯如圓形圈所示。主機將傳送新的服務埠與隨機亂數，加密後傳至給客戶端，客戶端解密後將收到新的服務埠號碼為 5001。

以下將介紹主機轉移服務埠時封包流向，圖 5.3 中，圓圈標示為啟動 Xwindow 之指令 startxwin。圖 5.4 中，所圈起來為標示執行檔案 ns routing.tcl。

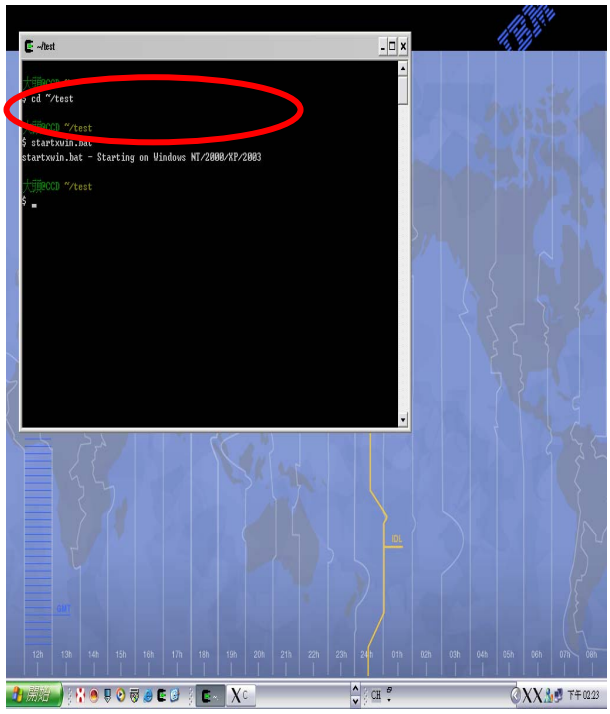

圖 5.3 轉移服務埠機制

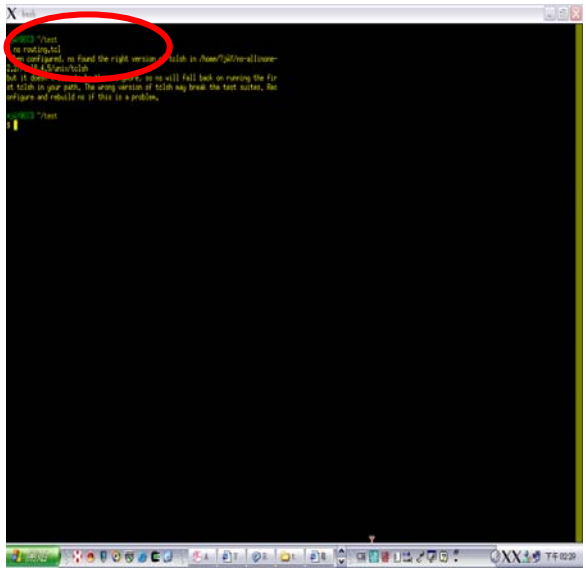

圖 5.4 執行檔案

在圖 5.5 中顯示正常提供服務時封包流向，而在圖 5.6 中顯示當執行轉移服務埠時，在 1.23 秒將會執行轉移服務埠，在圖 5.7 中當執行完轉移服務埠機制時，將會在 2.29 秒時恢復正常提供服務。

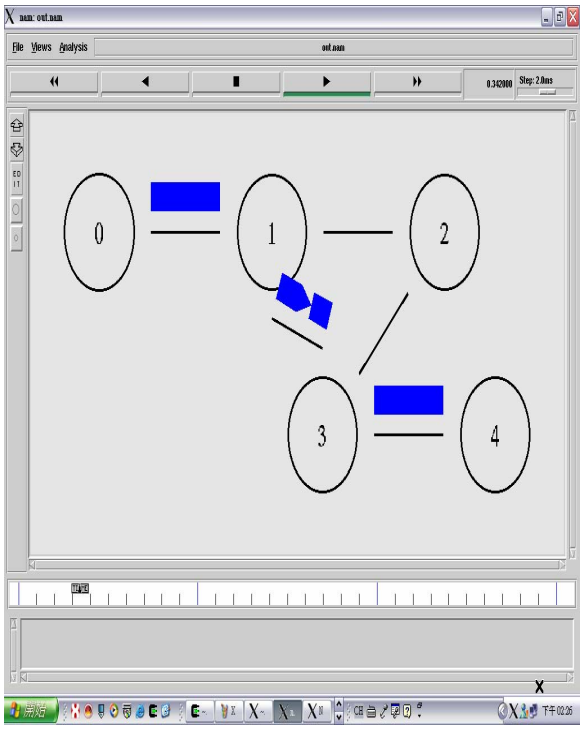

圖 5.5 正常封包流向

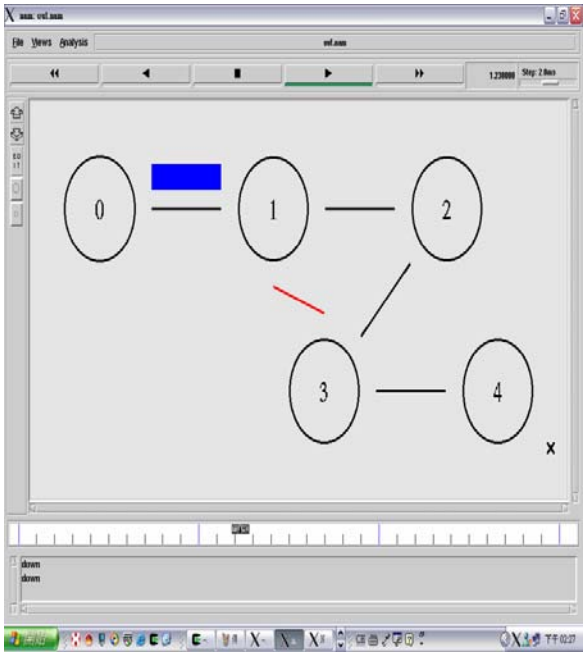

圖 5.6 執行轉移服務埠

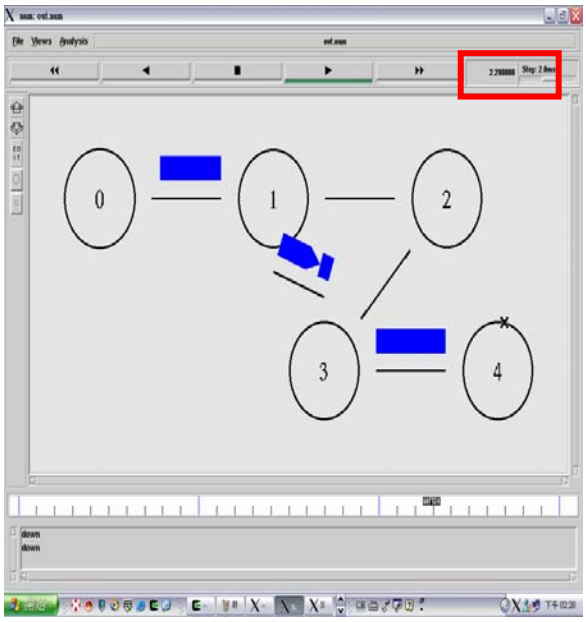

圖 5.7 恢復正常服務

(2)備援機制

以下將介紹第三種防護，圖 5.8 為執行備援機制檔案命令，首先在 Xwindow 下達指令 ns backup.tcl DV，將跑出網路架構畫面如圖 5.9 所示。圖 5.10 為主機正常畫面，並且會建立相關路由表，當主機 3 被攻垮時如圖 5.11，在圖 5.11 中 1 號主機至 3 號主機呈現斜線畫面表示已無法連線，圖 5.12 為主機 2 立即啟動備援機制。而另一攻擊畫面為圖 5.13，當主機 5 被攻垮時，圖 5.14 中主機 4 立即啟動備援機制，最後當主機 5 恢復正常後如圖 5.15 所示。

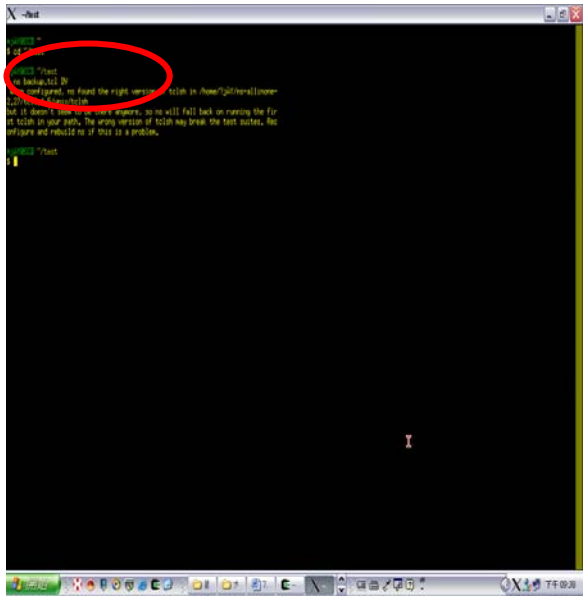

圖 5.8 執行備援機制檔案

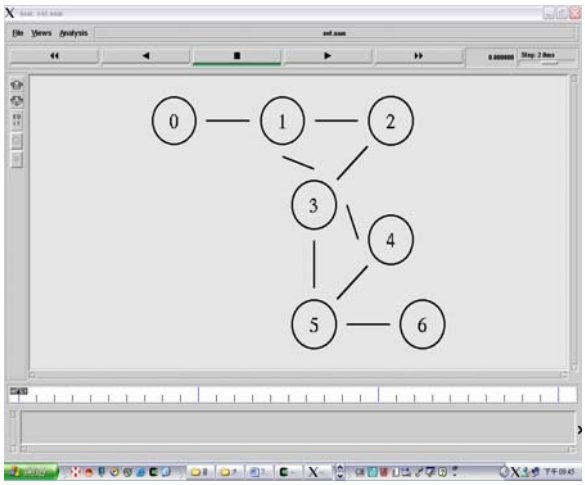

圖 5.9 網路模擬架構

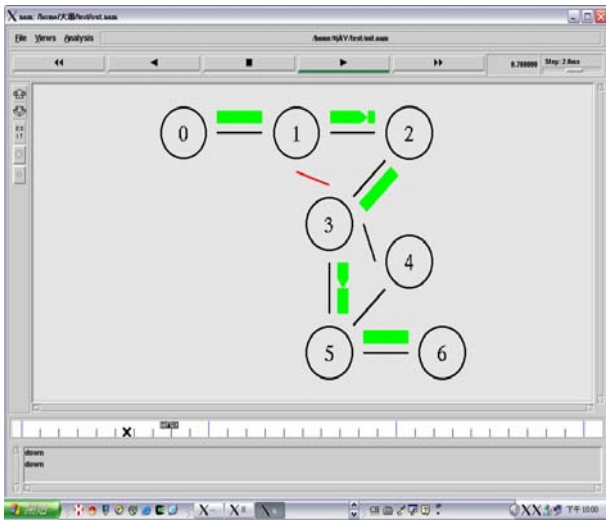

圖 5.12 主機 2 啟動備援機制

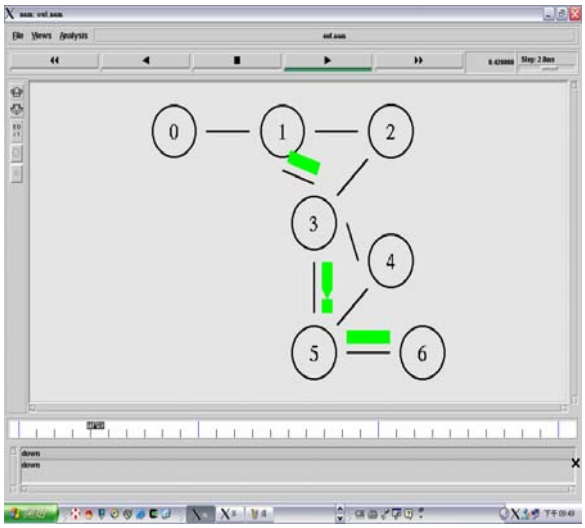

圖 5.10 正常主機畫面

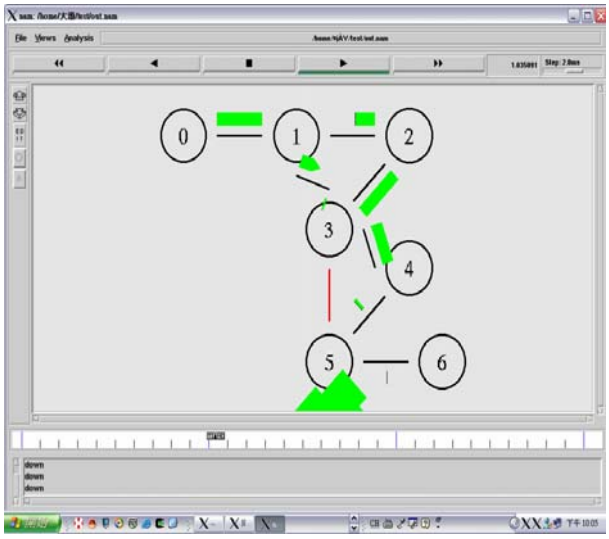

圖 5.13 主機 5 被攻垮

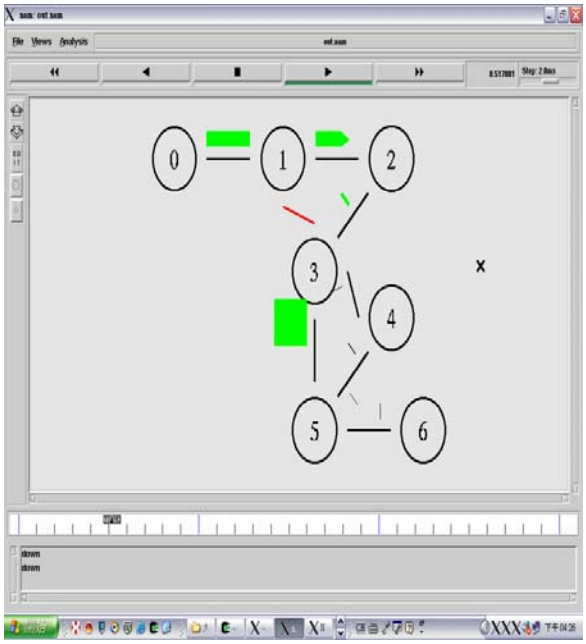

圖 5.11 主機 3 被攻垮

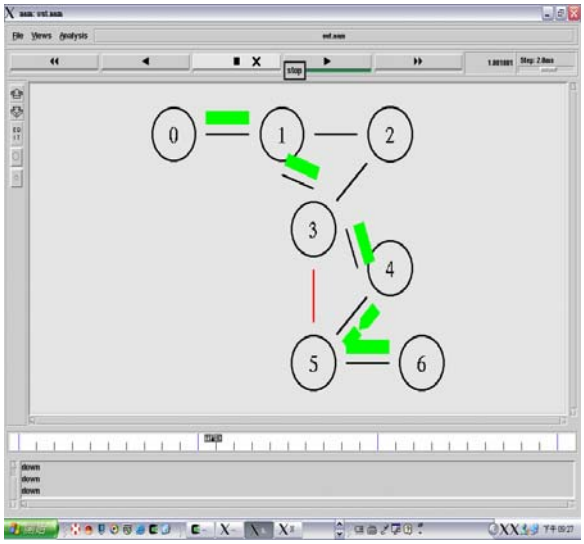

圖 5.14 主機 4 啟動備援機制

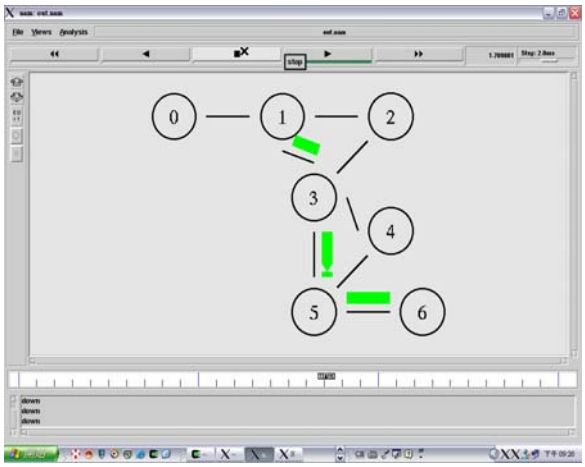

圖 5.15 主機 5 恢復正常

6. 結論

由於洪水式分散阻絕服務攻擊現行防禦機制無法抵抗合法使用者攻擊，並且當主機被攻擊致癱瘓時無法即時備援，因此本研究基於多代理人架構，並整合三種防護機制，第一種為基於橢圓曲線公開金鑰密碼系統來驗證使用者之合法性，其次當合法使用者攻擊時，將啟動轉移服務埠機制，使得當合法使用者大量送出封包攻擊時，此防禦機制可以轉移服務埠而繼續提供服務。最後為備源機制，當偵測主機被攻擊時，偵測代理人會通知兩側主機，並且選擇負荷量最小的主機即時備援。進而達成真正抵抗洪水式分散服務攻擊。

由本研究的安全性分析得知，此三種防禦的確能抵抗洪水式分散服務攻擊，雖然採用驗證機制，但延遲的時間並不影響效能，並且當受攻擊時，服務仍然是存活的，最後當主機被攻垮時，旁邊兩側之主機的確可以即時備援。

由於目前防禦機制尚有 IP Trace Back，但無有效方法，因為跨網域的追蹤機制有許多困難處，因此希望本研究將在未來結合來源端追蹤機制，更能有效阻擋洪水式分散阻絕服務攻擊。

致謝

本研究接受國科會研究計畫案 NSC 94-2219-E-212-001 資助，特此致謝。

參考文獻

[1]林楷勛(2004)。適用於保護應用伺服器以防止分散式阻斷服務攻擊之入侵保護系統。國立清華大學資訊系統與應用研究所碩士論文，新竹市。

[2]曹偉駿、吳正光(2003)，用行動代理人以抵抗分散式阻絕服務之入侵偵測機制,第十三屆全國資訊安全會議，頁 349-360。

[3]CERT Statistical Weaknesses in TCP/IP Initial Sequence Numbers, from http://www.cert.org/stats/index.html

[4]CERT/CC Statistics (1988-2006). Retrieved May 19, 2006, from http://www.cert.org/stats/cert_stats.html

[5]Barros, C., A proposal for ICMP trace back messages, Internet Draft. from http://www.research.att.com/lists/ietfitrace/2000/09/msg00044. html, Sept 18, 2000.

[6]Burch H., & Cheswick H. (2000). Tracing anonymous packets to their approximate source. *in Proc. USENIX Conference,* (pp. 319-327)

[7]Caelli, W., Dawson, E., & Rea, S., (1999). PKI, elliptic curve cryptography and digital signatures. *Computer & Security*, 18(1), 47-66.

[8]Chen, L., Longstaff, T., & Carley, K. (2004). Characterization of defense mechanisms against distributed denial of service attacks, *Computers & Security,* 23(8), 665-678.

[9]Chen, S., Tang, Y., & Du,W. (2006). Stateful DDoS attacks and targeted filtering. *Journal of Network and Computer Applications,* (Accepted).

[10]Chen, S., & Song, Q. (2005). Perimeter-Based Defense against High Bandwidth DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, 16(7), 784-799.

[11]Cubaleska B., & Schneider M. (2002). Detecting DoS attacks in mobile agent systems and using trust policies for their prevention. *The 6th World Multiconference on Systemics Cybernetics and Informatics*, (pp. 421-434).

[12]Douligeris, C., & Mitrokotsa, A., (2004). DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks,* 44(5), 643-666.

[13]ElGamal T., (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4), 469-472.

[14]Ferguson, P., & Senie, D., (2000). Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing agreements performance monitoring,

RFC 2827.

[15]Habib, M., Hefeeda, & Bhargava, B., (2003). Detecting service violations and DoS attacks. *in Proc. Network and Distributed System Security Symposium (NDSS '03)*, San Diego.

[16]Jansen, W. (2002). Intrusion detection with mobile agents. *Computer Communications,* 25(15), 1392-1401.

[17]Jurisic, & Menezes, A.J., (1997). Elliptic curves and cryptography. *Dr. Dobb's Journal*, (pp. 26-35).

[18]Kenney, Malachi, Ping of Death, January 1997, Available from http://www.insecure.org/sploits/ping-o-death.html

[19]Kargl, F. Maier, J. Schlott, & Weber, S., (2001). Michael Protecting Web Servers from Distributed Denial of Service Attacks. *In Proceedings International WWW Conference*(10), (pp. 150-162).

[20]Koblitz, N., (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(17), 203-209.

[21]Kemmerer R., & Vigna G., (2002). Intrusion Detection: A Brief History and Overview. *IEEE Computer Special Issue on Security and Privacy*, (pp. 27-30).

[22]Mirkovic J., Martin J., & Reiher P., (2001). A Taxonomy of DDoS Attacks and DDoS Defense Mechanism. UCLA CSD Technical Report CSD-TR-020018.

[23]Mell, P., Marks, D., & McLarnon, M., (2000). A denial-of-service resistant intrusion detection architecture. *Computer Networks*, 34(7), 641-658.

[24]Mirkovic J., Prier G., & Reiher P., Attacking DDoS at the source. *Proceedings of ICNP 2002*, (pp. 312-321).

[25]Miu, K. N., Chiang, H. D., & McNulty, R. J., (2000). Multi-Tier Service Restoration Through Network Reconfiguration and Capacitor Control for Large-Scale Radial Distribution Networks. *IEEE Trans. on Power Systems*, 15(3), 1001-1007.

[26]McCanne S., Floyd S., NS-2 Network Simulator from http://www.isi.edu/nsnam/ns.hml.

[27]Miller., V.S. (1986). Use of elliptic curves in cryptography. Advances in Cryptology:Crypto'85, Springer-Verlag, (pp. 417-426).

[28]National Bureau of Standards, (1977). Data encryption standard. *Federal Information Processing Standards Publication FIPS PUB 46 U.S. Department of Commerce*.

[29]Park K., & Lee H.,(2004) . A proactive approach to distributed DoS attack prevention using route-based packet filtering. *in Proc. ACM SIGCOMM*, (pp. 124-136).

[30]Snoeren, C., Patridge, L ., Sanchez, C., Jones, F., & Tchakountio, B., Schwartz, et al., (2002). Single-packet IP trace back. *IEEE Transaction on Networking*, (pp. 721-755).

[31]Schulba I., Krsu, & Kuhn M.., (1997). Analysis of a Denial of Service Attack on TCP. Proceedings of the 1997 IEEE Symposium on Security and Privacy, (451-468).

[32]Spafford, C., (2000). Intrusion detection using autonomous agents. *Computer Networks,* 34(4), 547–570.

[33]Savage, S., Wetherall, D., Karlin, A., & Anderson, T., (2001). Network support for IP trace back. *IEEE/ACM Transaction on Networking*, 9(3), 226–237.

[34]Steels, L., (1995). When are robots intelligent autonomous agents?. J*ournal of Robotics and Autonomous Systems,* 15(1-2), 3-9.

[35]Shakshuki, E., Luo, Z., & Gong, J., (2005). An agent-based approach to security service. *Journal of Network and Computer Applications*, 28(2), 183-208.

[36]Schnorr C.P., (1990). Efficient identification and signatures for smart cards, Advances in Cryptology: Crypto'89, Springer-Verlag, (pp. 339-351).

[37]Tsaur, W. J., (2005). Several Security Schemes Constructed Using ECC-Based Self-Certified Key Cryptosystems. *Applied Mathematics and Computation,* 168(1), 447-464.

[38]Vanstone, S., (1977). Elliptic Curve Cryptosystem-The Answer to Strong, Fast Public-key Cryptography for Securing Constrained Environments. Elsevier In*formation Securtiy Technical Report*, 2(2), 78-87.

[39]Wang, Y., Behera, S., Wong, J., G., Helmer, V., Honavar, L., Miller, Lutz, R., & Slagell, M., (2006). Towards the automatic generation of mobile agents for distributed intrusion detection system. *Journal of Systems and Software,* 79(1), 1-14.