基於 NetFlow 之大型網路蠕蟲偵測系統

王曠銘　　羅孟彥*　　楊竹星
國立中山大學資訊工程學系　正修科技大學資訊管理系*
{wangkm, csyang}@cse.nsysu.edu.tw, myluo@csu.edu.tw*

摘要

網路蠕蟲近年來已成為網際網路上一個重要的安全議題，網路蠕蟲不僅是對終端主機造成實質上的資訊安全危害，其在進行攻擊時所產生的大量封包以及連線，連帶的降低整個網際網路運作的效能。此外，在管理大型網路時需要監測此類問題乃必須要有一有效的機制，由於傳統使用封包擷取之方式，難以有效處理大型網路上的流量，而若以 SNMP 監測流量異常，又失之細節，需要網路管理者進一步分析，耗力費時。有鑑於此，本文提出一使用 NetFlow 作為分析資料來源之網路蠕蟲偵測系統，協助大型網路之網路管理者監測可疑之網路蠕蟲活動，並協助網路管理者辨識網路蠕蟲之種類，有效瞭解實際問題，加速網路管理者處理異常之速度。

關鍵詞：網路異常偵測、網路蠕蟲、NetFlow

1. 前言

隨著網際網路(Internet)快速發展，網路得以廣佈至各個家庭個人，乃至大型企業以及國家政府，隨著使用人口的增加，網路上的流量愈加龐大，所使用的通訊協定組成也愈顯複雜，不若發展初期時單純。而近年來，網際網路安全面臨的一個主要的威脅來自於具有透過網路快速散佈及自我繁衍特性的惡意程式(malware) — 網路蠕蟲(Internet worm)，其對於終端主機的安全造成危害：進行破壞主機資料、植入後門(backdoor) 或是消耗主機資源等惡意行為，除此之外，網路蠕蟲在進行感染時，所造成的大量封包和流量也對網路上負責交換網路流量的設備構成負擔，造成網路效能低落，甚至變相形成一種阻斷服務攻擊(Denial of Service)，這些都使得整個網際網路網路安全和運作受到嚴重的威脅，因而無論在業界或是學術界，網路蠕蟲的相關討論皆躍升為主要的網路安全議題之一。

在管理大型網路的實務經驗中，網路管理者在面臨網路蠕蟲的問題時，往往在第一時間感知的異常狀況是底下的使用者向管理者反應網路速度變慢，但構成網路速度變慢的成因具有許多種可能，有可能是設備異常或是路由異常等問題所造成，而若是由網路蠕蟲造成的，那也可能是因為網路蠕蟲造成的大量流量使得整個出口頻寬被塞滿，又或著是由其所產生的大量連線數，讓路由器(router)疲於應付，無法將整個路由工作由硬體完成，只好再將部分工作交給 CPU 處理，使得整個路由器的效能劇烈惡化，無法順利完成其轉送封包的工作，而管理者必須要透過種種管理經驗和網路分析才能得出實際原因所在，等待處理完畢，該異常現象往往已經使得網路癱瘓一段時間。

本研究使用 NetFlow 作為分析資料來源，實作一網路蠕蟲偵測系統，協助網路管理者在大型網路中監控網路蠕蟲異常現象，有效掌握網路蠕蟲之問題，以增進控管網路之安全性。

2. 相關研究

Weaver 等學者[11]分別對現有之電腦蠕蟲的探索目標方法(target discovery)、傳播的方式與管道(propagation carriers and distribution mechanisms)、觸發蠕蟲活動的方式(activation)、蠕蟲除了散佈之外具備的其他功能(payloads)以及蠕蟲作者的動機(motivation and attackers)進行歸納分類，本研究主要針對使用掃瞄(scanning)作為探索目標方式的網路蠕蟲進行分析。

目前用以偵測網路蠕蟲之方法主要有三類：1. 透過分析網路蠕蟲所產生之異常封包內容來偵測網路蠕蟲，目前以比對網路蠕蟲攻擊封包內容特徵為主流[5][8][9][10]　2.分析主機所產生的連線是否超過正常應用之連線數，或這些連線大多無法成功建立，來判定這些連線活動是否正常，或是由疑似網路蠕蟲掃瞄所構成之現象[1][3][7]　3.在未使用的 IP 或網段上部署監測點，任何連向這些無實質作用的連線皆視為異常連線，分析這些連線，用以判定是否為網路蠕蟲之攻擊[2][4][6]。這些方法各自依據網路蠕蟲活動時所產生的特性來進行偵測，其中 Lai 等學者同是以 NetFlow 作為分析網路蠕蟲活動之資料來源。

3. 系統設計與實作

在本研究中，我們實作一套基於 NetFlow 作為資料來源的網路蠕蟲偵測系統—FloWorM，對於 FloWorM 偵測到之警訊，可以採取告知網路管理者作進一步分析或協同網路管理系統(Network Management System, NMS)進行自動化的防衛以及處置。

3.1 偵測網路蠕蟲之異常連線行為

我們根據目前網路蠕蟲之行為歸納出兩項特徵作為主要偵測演算法之依據。網路蠕蟲為了感染新的主機，會快速產生大量連線去尋找(discovery)新的宿主，我們將其稱為”擴散”(spreading)現象，又我們假設網路蠕蟲對攻擊目標的資訊一無所知，在其任意產生攻擊目標的情況下，我們預期其所建立的連線有很高的比例會失敗，此時便產生一種”掃瞄”(scanning)的行為。

一般正常的使用者在使用網路的時候通常都會連向自己固定少數幾個特定偏好的網站，而與網路蠕蟲最大的差異在於，網路蠕蟲的目標在於不斷找尋新的潛在感染目標進行感染的行為，故其建立連線的新目標會不斷產生，而非特定在某幾個特定的主機上，我們將此行為稱為”擴散”，我們將”擴散”定義為：某 source IP 在特定時間區間 T 內，對特定 {protocol, dstPort} 嘗試建立連線的目標主機數量 SP 超過一定的門檻值 $SP_{threshold}$ 時，我們就將 source IP 標記為正在對 {protocol, dstPort} 進行”擴散”。例如：10.0.0.2 在三十分鐘內對超過 200 台主機的 80/tcp 進行連線，此時該相關的 flow 皆會被視為”擴散”異常之 flow。

然而”擴散”該網路行為非常有可能出現在一般正常用途的 proxy server 以及 NAT server 上，在大型網路中這是十分常見的服務形式，此種正常的服務與網路蠕蟲掃瞄最大的差異點在於，proxy server 與 NAT server 其產生的連線都是由其後所服務的一般的使用者所產生之連線，這些連線應多可以與連線目標正確建立起連線，進而在建立的連線上進行資料的傳遞，而網路蠕蟲所嘗試建立的連線則往往無法建立起成功的連線，因此針對此點我們必須加以區分出連線的成功與否，來加以鑑別這些正常的網路行為以及異常的網路行為。

常見的失敗連線(negative connection)種類有對方完全無回應任何封包、對方回應 TCP RESET 拒絕連線或對方/路由器回應 ICMP port/host unreachable 訊息，我們將遇到這些狀況的連線視為失敗連線，反之，若對方有回應任何資料，我們將其視為成功的連線，因此我們透過判定特定主機所產生之主動連線 flow 是否有相對應的回應資料 flow 產生，以決定該連線是否成功建立。

而當”擴散”出去的連線其中大部分是失敗的連線時，我們便將此現象稱為”掃瞄”，將其定義如下：當某 source IP 滿足”擴散”條件，且該 source IP 對特定 {protocol, dstPort} 之連線失敗主機數 F 佔總建立連線主機數 SP 的失敗連線比例 NR (negative rate) = F/SP 超過一定的門檻值 $NR_{threshold}$ 時，我們將其判定為”掃瞄”。例如：前例所提之 10.0.0.2 對 80/tcp 所建立的連線中有 85% 為失敗連線，其後相關之 flow 會被視為”掃瞄”異常之 flow。

本方法有兩個主要的臨界值參數，第一個為滿足”擴散”條件的 $SP_{threshold}$，第二是滿足”掃瞄”條件的 $NR_{threshold}$，對於 $SP_{threshold}$ 而言，我們將時間區間 T 依據路由器 flow 最長保留時間設定為三十分鐘，根據經驗我們將 $SP_{threshold}$ 設定為 200，理論上我們可以偵測到掃瞄速率在每分鐘 $SP_{threshold}$ / T 個連線以上的的網路蠕蟲，而一般網路蠕蟲的掃瞄速率通常皆高於每分鐘一百台主機以上，若要嘗試躲避這項條件的偵測，網路蠕蟲必須將自己的掃瞄速率降的很低(每分鐘 6.6 台主機的連線速率以下)，如此一來，網路蠕蟲的傳染速度勢必也會大幅受到影響，而網路管理者將擁有較充裕的時間去發現並處理該問題，除此之外，低掃瞄速率對於網路的影響也降低許多，對網路效能的衝擊較小。

而對於 $NR_{threshold}$，我們對幾個比較常見的服務進行分析，常見的 P2P 軟體 eMule/eDonkey 對於 4662/tcp 以及 4672/udp 的 negative rate 分佈約集中在 0.4~0.7 間。而 web server 常使用的 80/tcp 的 negative rate 分佈則呈現兩極化，negative rate 較低(<0.4)的團體組成大多為我們已知的 web proxy 和 NAT 的服務，而 negative rate 較高(>0.8)的團體可能由掃瞄或是網路蠕蟲攻擊所組成。

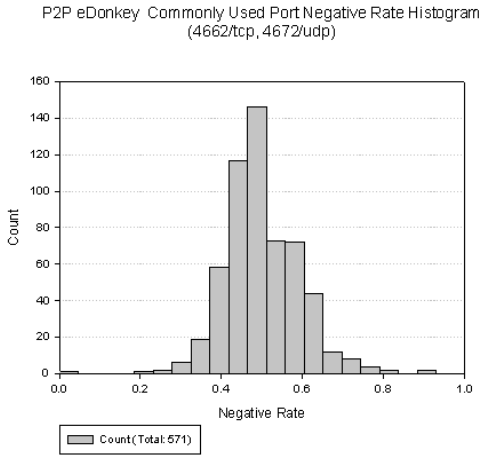

圖 1. Negative rate 分佈：eDonkey 預設 port

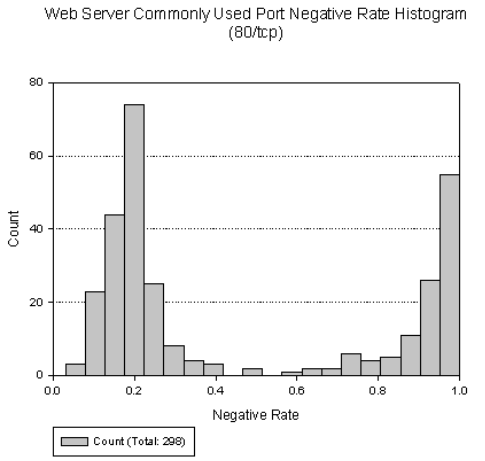

圖 2. Negative rate 分佈：80/tcp

理論上將 $NR_{threshold}$ 設定的越高將使得系統的

誤判率降低，但連帶使得偵測率下降，反之降低 $NR_{threshold}$ 雖然提升偵測率，但顯然也會提高誤判率，因此我們透過分析歷史警訊資料(參見圖 3)來決定系統之 $NR_{threshold}$，最後選擇 0.8 作為系統預設值，該值具有較理想的偵測率並兼具較低的誤判率。

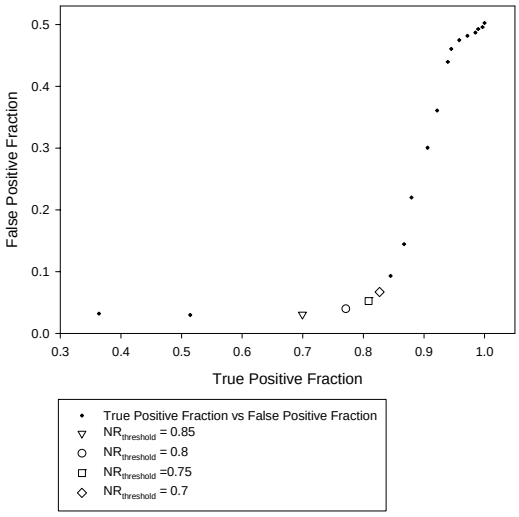

圖 3. $NR_{threshold}$ 與偵測率/誤判率之關係圖

然而在使用 NetFlow 分析 negative rate 時，尚會產生一個問題，由於 flow 通常會在路由器中保留一段時間，才會進行匯出之動作，因此會出現先前匯出的 flow 尚須一段時間才能取得對應的回應 flow 匯出，此時我們才能確定該 flow 代表的是一正常的連線。以下圖為例，一開始的 SP 值已經超過系統預設之 $SP_{threshold}$，而同時間，NR 值也超過 $NR_{threshold}$，已經符合"掃瞄"之條件，但事實上該主機為一正常的 web proxy，而經過一段時間後，開始收到對應的回應 flows，故 NR 值最後趨近於 0.2，屬於正常範圍($<NR_{threshold}$)。

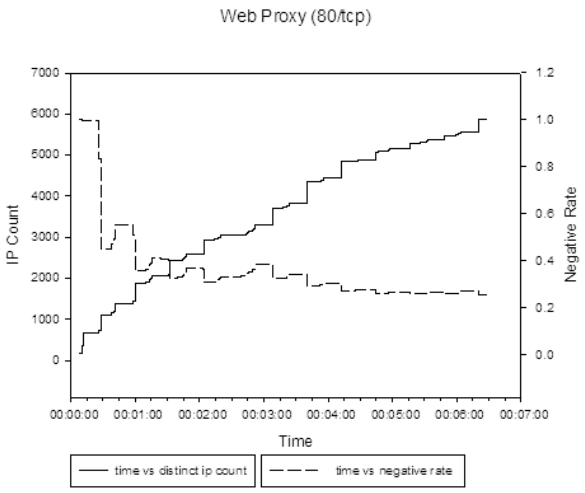

圖 4. 時間與 SP/NR 之關係圖

為了處理此一問題，我們將嘗試等待一段時間 holdingTime 後才承認 NR 趨於穩定，並參考 NR 值作為判斷是否為"掃瞄"之依據。對於網路蠕蟲的偵測而言，我們希望 holdingTime 能夠愈短愈好，如此對於網路蠕蟲才能盡早發現並且採取適當的防禦動作，但是過短的 holdingTime 會使得誤判率上升，因此我們使用固定時間區段的 NetFlow 資料分析 holdingTime 與 false positive 的關係(參見圖 5)，最後系統以 60 秒作為一個兼顧反應速度以及低誤判率的選擇。

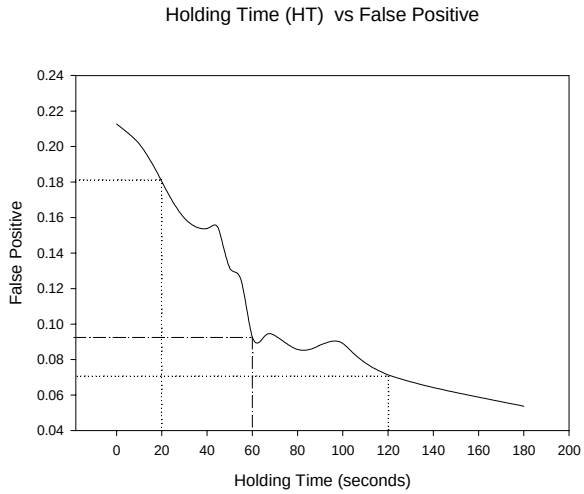

圖 5. holdingTime 與誤判率之關係圖

3.2 辨識網路蠕蟲之種類

當我們偵測出疑似網路蠕蟲所造成的異常掃瞄行為時，我們希望能進一步鑑定該異常是由何種網路蠕蟲所產生。傳統以封包特徵為基礎的入侵偵測系統中，透過比對封包內容的方式來判定攻擊種類，我們在此應用類似的概念，將網路蠕蟲各自特有的網路感染行為作為"網路特徵"(network signature)用以比對網路蠕蟲的種類。

網路蠕蟲通常攜帶特定的攻擊手法，透過攻擊特定的弱點服務作為入侵的手法，這些感染行為以透過掃瞄的方式尋找新宿主，是本系統主要的偵測目標，然而要完成整個感染行為不僅僅是送出攻擊封包，常見的其他行為還包含連向受害者的後門、向特定主機連線交換資訊、使用特定 port 傳送資料等等，這些由於都是透過網路作為資訊傳遞的管道，故我們在 NetFlow 上都可以觀測到這些行為，作為辨識的依據。

我們以 Blaster 做為例子，a 為感染 Blaster 的主機，b 為 a 正在進行感染的主機，a 首先會對 b 的 135/tcp 進行連線，此為 Blaster 的主要感染手段，也是 FloWorM 必須捕捉到的掃瞄行為，在連線建立成功後，a 會對 b 送出 exploit 進行攻擊，若成功將會在 b 的 4444/tcp 上建立一個後門，此時 a 會嘗試連向 b 的 4444/tcp 控制 b 透過 TFTP 自 a 的 69/udp 下載 Blaster 的本體程式回來並執行之，至此才完整整個感染行為。我們

透過這個範例將 Blaster 的 network signature 定義如下：

```
Blaster(a, b):
      a 連至 b 的 135/tcp，送出 exploit
      a 連至 b 的 4444/tcp，控制後門
      b 連回 a 的 69/udp，下載 Blaster
```

其他幾個知名的網路蠕蟲如 Slammer、Sasser 以及 Welchia 等，我們也可以同樣的方式，定義出它們的 network signature，作為辨識種類的方法。

該網路特徵辨識除了辨識網路蠕蟲種類外，亦可用以辨識幾個常見的 P2P 服務，以 eMule/eDonkey 為例，該軟體常見誤判的狀況為 peer 之間必須互相進行連線交換資訊，而各 peer 間使用的預設 port 為 4662/tcp 和 4672/udp，在對其他 peers 連線時，由於同時會產生許多連向 4662/tcp 或 4672/udp 的連線，造成看起來像是對 4662/tcp 或 4672/udp 掃瞄的假象，雖然我們可以以 negative rate 區隔出大部分此類現象與真正的掃瞄行為，但有時少數仍然會出現超過 $NR_{threshold}$ 的狀況，針對此一狀況，我們可加上 eMule/eDonkey 網路特徵辨識加以區別，我們對該 P2P 軟體的觀察為當 peer a 對 peer b 建立連線後，peer b 也有可能會對 peer a 主動建立連線，若觀察到此類模式，我們將該行為判定為 P2P，而非異常掃瞄行為，我們將 eMule/eDonkey 網路特徵定義如下：

```
eMule/eDonkey(a, b)
      a 連至 b 的 4662/tcp 或 4672/udp
      b 連至 a 的 4662/tcp 或 4672/udp
```

目前對於已知的 P2P 網路特徵辨識最大的問題點在於使用者可以自行修改其預設的 listen port，以 eMule/eDonkey 為例，使用者若將預設的 listen port 修改為非 4662/tcp 或 4672/udp，我們將難以觀測到” b 連至 a 的 4662/tcp 或 4672/udp”此一行為，該網路特徵也就無法成立，若該主機又恰好又滿足”掃瞄”條件，依舊會造成誤判，但我們加入 P2P 網路特徵的目的在於排除少部分的 P2P 誤判狀況，故此一情形對主要的偵測方法並不構成太大的影響。

FloWorM 會持續收集符合網路特徵的 flows，並將相關警訊標記上該特徵，當某警訊上標記的特徵比對網路特徵表格後，符合特定網路蠕蟲或是網路應用時，該警訊便被視為由該種網路蠕蟲或網路應用所產生。這與傳統的特徵比對方式本質上相同，因此若是新的網路應用流行或是網路蠕蟲發作，而我們系統中無此一網路特徵便無法進行辨識。

3.3 產生系統警訊與處置

最後，系統需決定何者為真正的網路蠕蟲行為，何者為非，目前 FloWorM 判定網路蠕蟲行為的三項條件為：

A.該{flow.srcIp, flow.protocol, flow.dstPort}之相關 flows 符合”掃瞄”現象

B.該{flow.srcIp, flow.protocol, flow.dstPort}之警訊記錄持續時間已超過 holdingTime 秒。

C.該{flow.srcIp, flow.protocol, flow.dstPort}之警訊記錄未符合已知 P2P 軟體之網路特徵。

當符合這三項條件時，FloWorM 即判定此為一網路蠕蟲行為，並呈現警訊資料給網路管理者或是通知網路管理系統進行處理。目前 FloWorM 除了將警訊輸出至一般的文字記錄檔中，也將警訊內容輸出至資料庫，並且透過 web 介面呈現。目前我們呈現的資訊如下表所示。

表 1 FloWorM 警訊欄位說明

欄位名稱	說明
Source Interface	來源主機之路由器介面編號
Source IP Address	來源主機之 IP 位址
Is Managed IP	是否為受控網路之 IP
Protocol	使用之通訊協定
Destination Port	目的主機所使用的埠號
Start Time	警訊起始時間
End Time	警訊終止時間
Duration	警訊持續時間
Scanning Rate	掃瞄速率(per minute)
Destination IPs Count	累計目的主機數量
Honeypot Hits Count	累計連至未使用 IP 位址數量
IANA Reserved IPs Count	累計連至 IANA 未配置 IP 位址數量
Network Signature Name	辨識網路特徵之名稱

4. 驗證方法與實驗數據

在實驗過程中，我們分別使用中山大學校內路由器 Cisco 7609 以及高屏澎區網路由器 Cisco 6509 的 NetFlow 作為資料來源並進行分析，而由於校內的環境以及問題點較容易掌握，在此主要提供使用校內路由器 Cisco 7609 的 NetFlow 分析的結果作一說明及驗證，底下將分別對於系統之 false positive 以及 false negative 進行闡述。

4.1 False Positive

對於異常偵測系統而言，讓網路管理者最為困擾的往往是假警報(false alarm)，也就是所謂的 false positive，這些假警報可能使得網路管理者白忙一場，或是最後因為假警報過多，使得網路管理者不信任這些警報，偵測系統視同虛設。此外，這對於抵禦網路蠕蟲而言尚還具有一個意義，由於網路蠕蟲的爆發時間極短，若是處理過程中需要透過人力介入，往往不及網路蠕蟲的散佈速度，因此若網

路蠕蟲偵測系統所產生的警訊誤判率極低，提供網路管理者一個高度可信的異常警訊，網路管理者就能採取完全自動化的機制以抵禦網路蠕蟲的爆發。

False positive 的部分，我們使用自動與人工並行驗證，我們首先在校園網路中未使用的網路區段部署 honeyd 以及部分主機上部署 Snort，並透過 NetFlow 監測三十個未分配的校內 class C 網段。在實驗期間內，我們總共在三十個未分配網段監測約七千五百個 IP 位址，以及三個已使用網段中監測約六十個 IP 位址，任何連到這些 IP 位址的連線或是讓 Snort 偵測出警訊的連線都視為可疑的連線，這些可疑連線都會被紀錄至資料庫中，當 FloWorM 偵測到異常的網路蠕蟲活動時，會比對資料庫中該警訊異常活動發作的時間區段內是否有相關紀錄，若存在相關資料，即表示警訊中的來源 IP 曾經嘗試掃瞄過這些 honeyd，抑或是是攻擊這些 Snort 所在的主機，我們將其標定為 true，若找不到相關記錄，則由人工逕行到該主機前進行分析，判斷該主機是否有進行網路蠕蟲感染的行為，若有將被標記為 true，反之，即為假警報，標記為 false。

除此之外，由於我們必須等待感染網路蠕蟲的主機掃瞄到這些監測點才會在當時有相關紀錄留下，部分警訊會由於當時剛好沒有任何監測點受到攻擊而判定為 false positive，因此我們增列一項 historical true，當我們查詢監測點的歷史紀錄發現有紀錄時則將其歸在這項之中。再者還有一個要考慮的特殊狀況就是一個網路蠕蟲倘若其攻擊的模式為：”對 port X 成功送出攻擊封包，才會繼續連向 port Y”，而對於監測點而言，攻擊 port X 的紀錄雖然會被紀錄到，但由於大部分的監測點(honeyd 以及未使用的網路區段)並不會和網路蠕蟲成功建立連線進行互動，故我們也無法採樣到連線至 port Y 的紀錄，但實際上網路蠕蟲在攻擊時，若剛好掃到一個區段內都有開啟 port X 的服務的話，接下來對 port Y 的行為也將類似對 port Y 作掃瞄，因此我們對於若無法找到 port Y 的紀錄，但找得到該主機對 honeyd 的其他 port 進行連線的紀錄，我們將此類警訊歸類為 related true。

針對 false positive 部分，我們收集了 2005 年六月底至 2005 年八月初之資料，約一個半月的資料，以一天為單位進行警訊驗證分析，共累計產生了 1848 筆警訊，我們對這些警訊進行驗證，結果如表 2 所示，我們的偵測正確率平均皆都在九成以上。

而這些誤判之中，大部分誤判來自於 Peer-to-Peer 軟體(eMule, eDonkey, BitTorrent)，少部分來自一台校內的防毒伺服器所廣播出來的 UDP 封包，雖然無法達到百分之百的正確率，但如果能再加強對於 P2P 的辨識以及白名單的建立，相信將有助於正確率的提升。

表 2 FloWorM false positive 驗證數據

False Positive I	False Positive II	False Positive III
3.13%	3.60%	5.00%

False Posivie I = F / T

False Posivie II = $(F+R) / T$

False Posivie III = $(F+R+H) / T$

- *F:* number of false alarms
- *R:* number of related true
- *H:* number of historical true
- *T:* number of total alarms

4.2 False Negative

對於 false negative 的部分，由於實際上整體網路上的活動難以掌握，所以對於取得整個網路上所有感染網路蠕蟲主機的確實清單十分困難，因此我們使用兩個方式來驗證這部分的資料，一是透過分析歷史資料的方式來測試這部分的數據，二是由網管或是使用者告知疑似被網路蠕蟲攻擊的時間以及來源 IP 和目的 port，透過 FloWorM 分析該時段的 NetFlow 歷史資料，若有輸出相關警訊及標定為 true positive，若無則標定為 false negative。

而 false negative 的部分截至目前為止，使用者以及網路管理者所通報的部分約十多個案例，FloWorM 皆有相關紀錄。此外，使用 FloWorM 對大仁技術學院 2005 年二月校內主機感染 SQL Slammer 時之高屏澎區網之 NetFlow 相關記錄進行分析，所產生的警訊交由該校網管進行比對，顯示皆無漏判以及誤判的狀況。

4.3 與同類型系統之比較

我們並實作 Lai 所提出之偵測演算法[3]，該方法同是基於 NetFlow 作為資料來源，作者提出一個透過離線分析固定時間區間的 NetFlow 資料以偵測網路上類似網路蠕蟲攻擊行為的方法，透過統計來源主機連向不同目的主機次數、連向保留 IP 位址次數、重複 flow pattern 次數和疑似對連續 IP 位址進行掃瞄之次數這些數值計算出一評估參數來判斷是否疑似感染網路蠕蟲，若高於特定的 threshold，則判定為網路蠕蟲感染，並且進一步將偵測出的可疑主機隔離於受控網路(managed network)之外。

我們在初步實驗中，直接引用該篇論文中的參數，但偵測結果只取出 high watermark (threshold) 的警訊，而資料來源我們將同取任意十分鐘中山校園網路骨幹路由器的 NetFlow 資料作為兩者相同輸入資料來源，並以前述同一驗證方法比較兩者之 false positive（參見表格 1）。

表 3 與[3]偵測結果之比較

Date / Time	Total Alarms		True Positive		False positive	
	FWM	Lai	FWM	Lai	FWM	Lai
7/7 23:00	14	42	14	10	0%	76.2%
7/8 11:00	23	53	21	20	8.7%	62.2%
7/9 17:00	16	52	16	16	0%	69.2%
7/10 14:00	16	50	16	15	0%	70%
Total/Avg.	**69**	**197**	**67**	**61**	**2.9%**	**69.0%**
FWM: FloWorM						

比較結果顯示 FloWorM 的 false positive 極低，而 Lai 的方法對於常見的 P2P 軟體誤判率過高，使得該方法的 false positive 過高，此外，FloWorM 也偵測到數個 Lai 的方法遺漏的警訊，其有些落在 Lai 的 low watermark 區，有些則完全沒有偵測到，這些都形成該方法的 false negative。因此，FloWorM 無論在 false positive 或 false negative 的表現上都較 Lai 的方法為佳。

而我們認為 Lai 的方法誤判過多的原因來自於其未考慮到網路蠕蟲之掃瞄大多會產生失敗的連線，而只考慮連向多台主機的"擴散"連線行為，因此容易造成對 P2P、NAT Server 以及 Proxy Server 等具有此類似特性之服務的誤判。

5. 結論與未來工作

在本論文中我們實作了一個以 NetFlow 為資料來源的網路蠕蟲偵測系統，與過去的系統比較而言我們做出了以下幾點貢獻。

克服處理大型網路流量的瓶頸：使用 NetFlow 作為分析的資料來源的 FloWorM，在面臨流量龐大的大型網路時，能較傳統透過擷取封包並比對封包特徵的方式來得有效率，而且部署容易，只需要將路由器的 NetFlow 匯出(export)至 FloWorM 即可進行即時分析與偵測。

高正確率以及偵測率：FloWorM 使用 NetFlow 分析連線是否成功建立之概念來監測網路蠕蟲活動，相較過去同樣基於 NetFlow 作為資料來源之網路蠕蟲偵測方法，大幅減少了對一般 proxy、NAT 和 P2P 軟體的誤判，進而提供較高的正確偵測率。

透過網路特徵來辨識網路蠕蟲：FloWorM 透過監測網路蠕蟲的特殊網路行為，進而達到辨識網路蠕蟲種類的目標，加強了偵測結果的信心值，以及增進網路管理者辨識網路蠕蟲問題的速度。

然而由於 NetFlow 提供的資料量少，可以達到分析速度較快且可分析資料量較大的優點，但這無疑也是一把兩面刃，在精確度上亦有其需顧慮的問題存在，我們透過 NetFlow 分析網路蠕蟲攻擊之過程中，常會對於一些 P2P 等較特殊的應用程式或通訊協定構成誤判，若能將 NetFlow 所提供的資料，盡可能針對可疑的網路流量建構出線索以供分析之用，進而排除或是能界定這些特殊服務，將能達到更為精確的判斷結果。

未來我們希望能透過雙向 flow 的一些資料如進出封包數、進出流量和進出 flow 數等衍生出區分異常與正常流量的一些規則，以加強我們的判斷能力。此外透過分析歷史資料我們希望可以標定出哪些主機扮演著伺服器的角色，而哪些扮演著客戶端的角色，透過角色的網路行為模式加以辨識其網路行為是否有所異常，以期達到更理想的偵測能力。

致謝

本論文承蒙國科會計畫:

NSC 94-2622-E-110-008-CC3 之補助，特此致謝。

參考文獻

[1] Masaki Ishiguro, Hironobu Suzuki, Ichiro Murase, Hiroyuki, "Internet Threat Detection System Using Bayesian Estimation," 16th Annul FIRST Conference on Computer Security Incident Handling, 2004.

[2] Christian Kreibich, Jon Crowcroft, "Automated NIDS Signature Creation using Honeypots", Poster paper, SIGCOMM, 2003.

[3] Shou-Chuan Lai, Wen-Chu Kuo, Mu-Cheng Hsie, "Defending against Internet Worm-like Infestations", the 18th International Conference on Advanced Information Networking and Applications, 2004.

[4] David Moore. "Network telescopes: Observing small or distant security events", Invited Presentation at the 11th USENIX Security Symposium, 2002.

[5] Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Proceedings of the 7th USENIX Security Symposium, 1998.

[6] Niels Provos, "Honeyd: A Virtual Honeypot Framework", Proceedings of the 13th USENIX Security Symposium, 2004.

[7] Stuart Schechter, Jaeyeon Jung, Arthur W. Berger, "Fast Detection of Scanning Worm Infections", 7th International Symposium on Recent Advances in Intrusion Detection, September 2004.

[8] Sumeet Singh, Cristian Estanm, George Varghese, Stefan Savage, "Automated Worm Fingerprinting", 6th Symposium on Operating Systems Design and Implementation, 2004

[9] Snort - the de facto standard for intrusion detection/prevention, http://www.snort.org/.

[10] Ke Wang, Salvatore J. Stolfo, "Anomalous Payload-based Network Intrusion Detection", 7th International Symposium on Recent Advances in Intrusion Detection, 2004.

[11] Nicholas Weaver, Vern Paxson , Stuart Staniford , Robert Cunningham, "A taxonomy of computer worms", Proceedings of the 2003 ACM workshop on Rapid Malcode, 2003.