利用分散式入侵偵測與回應系統防治網蟲之入侵

陳奕明
中央大學資訊管理系
中壢市五權里 2 鄰 38 號
cym@cc.ncu.edu.tw

李勁頤
中央大學資訊管理系
中壢市五權里 2 鄰 38 號
aphyr@mis.mgt.ncu.edu.tw

摘要

2001 年 8 月， CodeRed 網蟲席捲全球網際網路，而緊隨而來的幾隻網蟲也在全世界造成了嚴重的災情，可見網蟲已經成為全球網路安全的嚴重威脅。不幸的是，目前我們並未見到一套好的防護系統可以抵禦網蟲的入侵。有鑑於此，本文特別提出一套分散式入侵偵測與回應系統，來偵測與抵禦網蟲的入侵。這套分散式入侵偵測與回應系統的主要特色包括 (1)整合 switch、router 等網路設備，以增強反制能力，(2)利用 graph engine 來進行關聯分析，以提昇對於網蟲的認知能力，(3)採用動態的回應機制，以加快回應速度，及對抗各不同種類型的網蟲。

論文中我們分別從(i)如何防治網蟲對區域網路內或外的攻擊，和(ii)如何防治網蟲由網際網路向內攻擊兩個角度來討論此分散式架構的應用。最後我們也說明實作的方式。我們相信這套分散式入侵偵測與回應系統，將能有效提昇對於網蟲入侵的抵抗能力。

關鍵詞：入侵偵測系統、網蟲、CodeRed Worm

一、前言

自從 1988 年 Morris Worm 在全球造成重大危害後，網蟲(Internet Worm)就成為網際網路上揮之不去的夢魘。從 1998 年開始盛行的 Email Worm[17]，如：Happy99、Melissa、Love Letter、Navidad 等，到 2000 年底和 2001 年初陸續出現的幾隻網蟲，如：Ramen[4]、1i0n Worm[22]、Sadmin/IIS Worm[5]，都迅速地在網路上造成廣泛的危害。而 2001 年中，CodeRed Worm[6]正式出現，根據 CAIDA 的分析報告[3]，CodeRed Worm 在短短的幾天內就感染了全球超過幾十萬台的主機。

由上述的網蟲入侵事例中，我們可以了解到，未來網蟲勢將成為網路攻擊的主流，許多漏洞的攻擊程式都將被包裝成網蟲的形式，用以發動大規模的入侵活動，身為網路的管理人員，我們必須開始思考如何抵禦網蟲的攻擊。

由於網蟲大多會利用合法的通訊管道進行入侵(如：CodeRed Worm 利用 80 port、http 協定)，傳統封包過濾(packet filter)的防火牆顯然無法抵禦網蟲的攻擊。入侵偵測系統雖然可能可以發現網蟲的蹤跡，但由於網蟲迅速傳播的特性，當網管人員看到相關的警示時，往往已經為時已晚，只能進行一些善後的處理工作。顯而易見地，傳統的防禦系統並不足以防範網蟲的入侵，因此在本文中，我們將提出一套新的防禦架構，來幫助我們儘早發現網蟲的蹤跡，並立即進行反制。

在接下來的文章中，我們將在第二節回顧現有的入侵偵測與回應系統，了解他們對於網蟲的偵測與防治能力；在第三節我們將將研究各種網蟲的傳播方法，以了解應該如何阻止網蟲的擴散與入侵；在第四節我們將提出一個分散式的網路入侵偵測與回應架構，用以防治網蟲的入侵；第五節說明系統實作：最後我們將在第六節作一簡短結論與未來研究方向以供後續研究者參考。

二、相關研究回顧

由於目前所發現的幾隻網蟲都是利用已知的漏洞進行攻擊(參見表一)，所以只要是設定良好的 misuse 網路入侵偵測系統應該都有能力偵測到這些網蟲所使用的攻擊方法。

表一、重要網蟲出現時間與所使用的漏洞

網蟲名稱	出現時間	使用漏洞 (公佈時間)
Ramen	2001/01/19	Rpc.statd (2000/07/21) Wu-ftpd (2000/06/23) LPRng (2000/12/12)
Lion	2001/03/28	Bind Tsig(2001/01/29)
Sadmin/IIS	2001/05/08	Solstic Sadmind (1999/12/24)
CodeRed	2001/07/19	IIS Indexing Service DLL (2001/06/18)
Nimda	2001/09/18	MS IIS Vulnerabilities (2000/10/10~2001/05/15) IE MIME Attachment Executable(2001/04/03) CodeRed Worm Back-door (2001/08/04)

不過由於大多數 misuse 網路入偵測偵測系統，於設計時並未考量到網蟲會大行其道，所以他們雖有能力偵測到網蟲所使用的攻擊方法，但由於對「網蟲」的認知能力有限，這些系統只知道發生了網路攻擊事件，卻無法判斷是不是網蟲所為，也難以做出適當的回應。

由此可見，想要偵測出網路上是否有網蟲在流竄，必須具備對於入侵事件的報告進行基本綜合分析、判斷的能力。以免費軟體 Snort[23] 為例，可利用加裝類似 Portscan Detector 的模組，透過數量累計的方式(如：在十秒內發現五次相同攻擊，視為網蟲)來推測所偵測到的網路入侵事件，是不是由網蟲所引起。

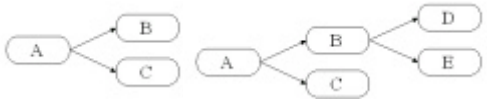
不過利用單點式的 misuse 網路入侵偵測系統來偵測網蟲，最大的缺點就是它必須被動的等待網蟲去重覆攻擊到同一網段，早已失去偵測的先機；再者，利用這種方式偵測網蟲，所能蒐集到的資訊其實非常有限，例如這類方法對於網蟲到底如何蔓延與整個網路受感染的情況就缺乏整體的了解。

在分散式入侵偵測系統方面[16][18]，大多數系統的設計目的就是要增加對於攻擊行為的認知能力，所以都具有最基本的綜合判斷能力。例如：Purdue 大學於 1995 年提出的 Autonomous Agents[18]，Joseph 等人提出的 Distributed Autonomous Agents[16]，都採用了 suspicious -level (或 alert-level)的概念，來加強對於分散式攻擊的偵測能力。他們的基本想法是分散於各處的 agents，若偵測到可疑事件，會透過廣播的方式通知其他 agents，所有收到緊告訊息的 agent 都增加自己的 suspicious-level，當 suspicious-level 超出警戒值則正式發出入侵事件的警示。Joseph 等人所定義的 alert-level = danger*transferability，更有助於及早發現網蟲的入侵。

很顯然的，若利用上述學者提出的 Distributed Autonomous Agent 來偵測網蟲，會比單點式的入侵偵測系統來得有效率，但由於這類的系統並沒有負責協調控制的中心，僅依靠廣播訊息相互溝通，若想由紛亂的警訊中整理出網蟲入侵的完整輪廓仍相當的困難。而且當網路遭受網蟲大規模入侵時，這類系統若沒有良好的資料縮減機制(data reduction)，傳遞於各 agents 之間的訊息將造成網路沉重的負擔，甚至拖累系統的效能。

而 AFFID[14]、EMERALD[21]或 Lighted Weight Agents[12]和 MAIDS[13][24]則提出了較完整的架構來說明如何統整分散於各處的入侵偵測系統。MAIDS 甚至利用 SFT、CPN 等工具，來說明如何由入侵偵測系統的需求分析與設計來自動產生所需的 agent，並構建可偵測分散式攻擊的系統。這些努力都有助於提昇對網蟲的偵測能力，不過由這些系統當初都不是為了偵測網蟲所設計，想將這些系統用於

網蟲的偵測 ，並了解網蟲蔓延的狀況，仍需要進行大幅的調整。以 MAIDS 為例，即需要由系統分析、系統設計與 agent 開發等流程，相當沒有彈性，不如以應付日新月異的網蟲。



(a) (b)

圖一、GrIDS 所建立的 Worm Graph

在眾多分散式入侵偵測系統中，對網蟲認知能力最好的當屬 UC Davis 的 GrIDS[24]，因為該系統原本的設計目的就在於偵測網蟲。該系統的偵測理念，主要著眼於網蟲擴散的天性。GrIDS 利用圖形表示法來描述網路中主機的活動狀態，例如：當 GrIDS 收到 A 入侵 B、C 的報告時，就會產生如圖一(a)的圖形。

若這是這是網蟲的攻擊，它應該會在短時間內繼續向外擴散，如網蟲會由受感染 B 主機，繼續向 D、E 展開攻擊，當 GrIDS 觀察到這個現象，也會將 B 攻擊 D、E 的行為表示成圖形，並依據特殊的圖形合併規則將兩個圖形合併，如圖一(b)。隨著網蟲不斷向外傳播，圖形也會越來越大，當圖形的深度或分支度超過警界值時，GrIDS 就發出警告。

我們可發現 GrIDS 所採用的方法，除了可以快速偵測出網蟲，更重要的是偵測結果可以很自然地描繪出網蟲傳播的模式，和目前網路遭受感染的狀況。此外 GrIDS 也提出了階層式圖形的表示方法，可以有效的壓縮網蟲入侵時產生的大量事件。

雖然 GrIDS 非常適合用於網蟲的偵測，但由於其並沒有一套好的回應機制，可用於阻止網蟲的繼續傳播，所以 GrIDS 用於防治網蟲擴散，能夠發揮的作用仍相當的有限。

面對網蟲驚人的傳播與攻擊能力，我們的防護系統不但必須快速偵測出網蟲的存在，還要能馬上做出適當的回應，在網管人員有時間處理前，就有效地壓制住網蟲。雖然已經有許多入侵偵測系統提供提供即時反制的能力，但由於入侵偵測系統的誤報率始終無法降低，常常使網管人員怯於採用。而且目前市面各種入侵偵測系統所提供的反制機制，往往過於簡單而無法產生效果。以入侵偵測系統最常提供的 TCP Reset 的功能為例，由於整隻 CodeRed 網蟲的大小不過 3K，只需 2~3 個封包就可完成攻擊，TCP Reset 就很難成功阻止 CodeRed 網蟲的攻擊。

有鑑於此，有愈來愈多的研究在探討如何提供更好的反制機制，例如：NAI Labs 的 Security Agility[20]，企圖為軟體模組加入安全認知能力，以支援執行期間動態安全政策的調整(runtime security policy changes)，避免軟體因為安全政策的改變而發生錯誤。而 IDIP[9]和 CITRA[10]則將焦點集中在如何能準確地反追蹤出入侵者，以降低即時回應所可能造成的風險。Curtis 等人[8]則企圖利用智慧型代理人程式來建構更好的回應機制，其特點在於系統會根據之前回應行動的成功與否，不斷調整回應策略。遺憾的是，這些研究大都仍在理論探討的階段，對於偵測與反制機制，尚未見到有具體的系統出現。

由上述的文獻回顧中我們可以了解到，要能有效防止網蟲的散佈，必須要能準確地找出受感染的主機，還要能彈性調整回應策略以對付各種不同的網蟲。

在下一節中，我們將研究各種網蟲的傳播方法，以了解該如何阻止網蟲的擴散與入侵。

三、網蟲的傳播方法分析

依據對過去幾隻 Worm 的了解，我們可以發現 Worm 的傳播方法通常分成四大類：

第一類：將自己夾藏在攻擊當中夾帶過去。例如 CodeRed Worm 的做法就是將整隻 Worm 夾藏在 Buffer Overflow 的攻擊碼中，一起送到受害者的電腦。

第二類：在成功攻入受害者的電腦後，反

向連回攻擊者端，將整隻網蟲完整下載的回來。例如 ramen worm 會先在攻擊者的電腦架設一個小型的 web server 在成功攻入受害者的電腦後再利用 lynx 反向連回下載 ramen.tgz。

第三類：透過攻擊在對方電腦留下後門或開啟特定 Port，再透過所開的後門或 Port，進行後續的攻擊。例如 Sadmin/IIS Worm 會在受害者的電腦開啟 port 600，透過該 port 對受害者電腦進行操控，並修改在 root 的 home 目錄下加入 .rhosts 檔，解除受害者端的 Access Control，再利用 rcp(remote copy file)將整隻 worm 上傳到受害者電腦。

第四類：在成功攻入受害者的電腦後，再利用如 tftp 等 UDP 協定，將整隻網蟲完整上傳或下載回受害者的電腦。 如 Nimda worm 其中一種傳播方式，就是透過 IIS 的漏洞或 CodeRed Worm 留下的後門對受害者的電腦下指令，指示受害者透過 tftp 反向回到攻擊者端下載相關程式。

依據這四種不同類型的攻擊方式，我們可擬定不同的防守策略。

關於第一類型的攻擊方式，由於攻擊者與受害者間只會建立一次網路連線，所以要防止網蟲擴散，只能針對這個攻擊連線進行處理，最常採用的方式是透過 TCP Reset 的封包來中斷該連線，不過這個方法不見得能成功。以 CodeRed Worm 為例，整隻網蟲的大小不過 3K，只需 2~3 個封包就可完成攻擊，所以利用 TCP Reset 的方式很難成功阻止 CodeRed Worm 的傳播。

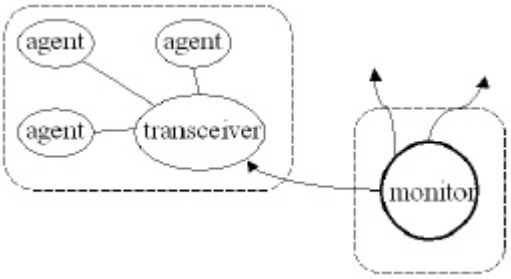
而關於第二和第三種攻擊方式，由於攻擊者和受害者之間還會建立其他網路連線，所以我們會有比較多的機會可以阻止網蟲的攻擊。例如我們可以建立動態的 TCP Reset 規則：當發現 A 機器對 B 機器發動攻擊後，立即阻斷所有 A 與 B 間的任何連線與阻止 A、B 間建立新的連線。由於在啟動動態 TCP Reset 規則後，可以無須解析封包的內容，就直接發出 Reset 封包，所以成功的機率將會比原本發現問題連線再中斷的方式來得高；而且根據經驗採取第二、三種攻擊模式的網蟲體積通常較大，也會花較久時間傳輸，所以防守者會有比較高的機率能成功中斷他們之間的連線。

第四種類型的攻擊方式，雖然於攻擊完成後，仍會有額外的網路流量用以傳輸網蟲本體，但由於所使用的是 UDP 協定，TCP Reset 並無法加以阻止。所以對於這類網蟲所能採用的回應方式類似第一類網蟲，只能利用 TCP Reset 來中斷攻擊連線。

四、分散式網路入侵與回應偵測系統

4.1 系統架構

根據前面的討論，我們提出了一個整合性的分散式入侵偵測與回應系統。該系統採用了類似 AAFID[14]的架構，主要包括 agent、transceiver、monitor 三大元件，如圖二。



圖二、系統基本元件

其中 agent 和 transceiver 是安裝在分散於各地的主機上，這些安裝 agent、transceiver 的電腦我們稱之為 local sensor，每台 local sensor 會有一個 transceiver 和數個 agent 程式。而 monitor 則是負責管理這些 local sensor 的程式。關於 monitor、transceiver 和 agent 的功能與特性，分述如下：

(1)agent：

agent 主要有三種類型，checker、log checker 或 responser。

checker agent 主要負責執行特定的檢查任務，如檔案完整性檢查等功能，並將結果報告

給 transceiver。log checker agent，主要負責整合本系統與其他安全系統，整合的方式主要是透過讀取其他安全系統的 log，進行格式轉換後報告給 transceiver 處理。responser 則是接受 transceiver 的指示，執行本系統各項回應機制，諸如：reset TCP session、透過 SNMP 重新設定 Switch、路由器與防火牆等設備。

(2)transceiver：

主要負責啟動、停止、執行與管理 agents，它必須接收各 agents 所報告的訊息，並依據簡單的研判規則進行處理，然後將精簡後的訊息報告給 monitor 進行處理。

此外 transceiver 會定期與 monitor 溝通，以確定該 local sensor 一切正常，並接受 monitor 的指示對 local sensor 進行相關設定。

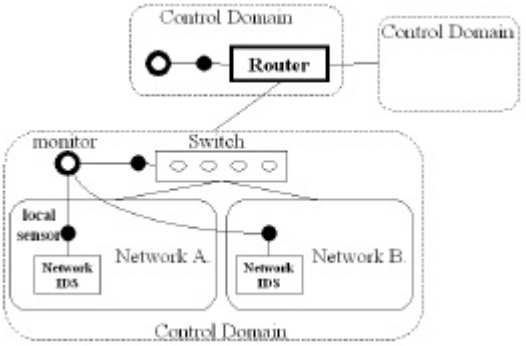
由於 local sensor 通常不是負責網路安全的專屬電腦，所以在設計上 agent 和 transceiver 的功能會儘量簡單，像 transceiver 所使用的研判規則，可能只是簡單的數量統計，如只單純計算在過去十秒內收到幾次異常事件報告。

(3)monitor：

為此系統的主控站，會檢視各個 local sensor 的狀況，並加以管理。此外 monitor 還會將各 local sensor 提出的報告進行關聯分析，依據分析的結果做處理。

monitor 為此系統的核心，相較於 transceiver 它會使用功能較為強大的關聯分析引擎，例如 UC Davis 的 graph engine。

我們可以利用上述分散式架構來整合其他安全設備，以建構足以防治網蟲擴散的防禦系統，參見圖三。在圖三中我們在每個區域網段都至少會有一台 local sensor，這些 local sensor 會與 Network-IDS 溝通，以了解目前該區域網段的狀態，此外還有部份 local sensor 是負責控制 Switch 或 router 等網路設備。數個 local sensor 將與一台 monitor 共同組成 control domain，由這台 monitor 負責協調指揮各 local sensor 的運作。



圖三：防禦架構示意圖

為了能確實阻止各種網蟲的擴散，我們認為必須在偵測的過程中就開始干擾網蟲，待後來蒐集更多資訊後，能更準確判斷出網蟲的類型後，再逐漸調整我們的回應策略。

所以我們將在一開始就採用較積極的回應手段，並不斷評估回應的成效，調整防禦策略。我們共定義了四種不同的危險等級，分別代表於偵測與防治網蟲過程中的不同狀態，在不同危險等級中將採取不同的回應措施，各危險等級所代表的意義如表二所示：

表二：四種網蟲入侵的危險等級

危險等級	代表意義
第一級	偵測到網路攻擊事件
第二級	發現網路攻擊事件有擴散的現象，確定這是網蟲在活動
第三級	第二級的回應機制無法確實發揮效用，採取更強烈的反制措施
第四級	受網蟲感染的主機已經太多，必須犧牲部份主機(中斷其網路服務)，以維持防護系統能繼續正常運作

本節接下來的部份，我們將依照網蟲攻擊的來源、方向分成兩部份來討論，首先介紹如何防治網蟲在區域網路內的交互感染與由區域網路向外攻擊，再介紹如何防治由網際網路向區域網路的攻擊。

4.2 防治網蟲在區域網路內或向外的攻擊

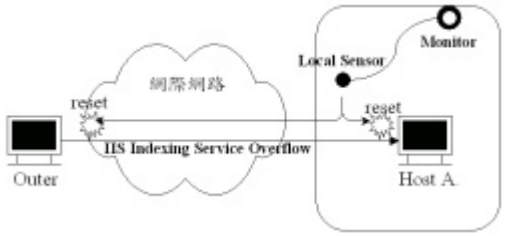
為了防治網蟲於區域網路內的活動， 於不同的危險等級，所將採取的回應措施也不

同，整理如表三：

表三：防治網蟲在區域網路或向外的攻擊

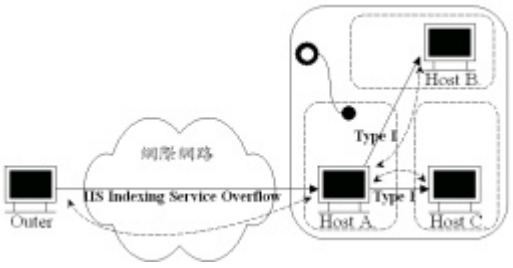
危險等級	可採取的回應措施
第一級	TCP Reset
第二級	Dynamic TCP Reset
第三級	Arp Spoofing
第四級	Switch block + Arp Spoofing

接下來我們將以 CodeRed 入侵為例，來說明我們的系統如何防治網蟲的入侵。如圖四所示：



圖四：採用第一級回應方法 TCP Reset

當 A 遭受 Outer(已感染 CodeRed 網蟲的外部機器)利用 IIS Indexing Service Overflow 攻擊時，我們系統會立即偵測到，並企圖利用 TCP Reset 阻止該攻擊的進行，若 TCP Reset 能成功阻止該攻擊，網蟲就無法繼續由 A 向外擴散。



圖五：網蟲由 Host A 繼續向外擴散

若 TCP Reset 無法成功阻止 IIS Indexing Service Overflow 的攻擊，A 將被網蟲感染並繼續向外攻擊。如圖五所示，A 將繼續利用相同的攻擊手法攻擊 B、C。當我們觀察到這種情形時可以知道，我們所採用的 TCP Reset 回應方法並無法有效防禦 IIS Indexing Service Overflow，而且由網蟲自 Outer 經過 A，並再度擴散到 B、C 這種行為模式，判斷這應該是有網蟲在活動。所以該 control domain 的 monitor 會立即採取第二級的回應機制，並將這個訊息報告出去，讓其他區域也可以立即採用類似的防禦手法。

由上一節對於網蟲傳播方法的分析中，我們知道，A 在被網蟲攻擊成功後，可能會被埋設後門，讓 Outer 能繼續連上 A，A 也可能會反向連回 Outer 將完整的網蟲下載回來(如 Ramen、Nimda 等)，如圖五中的虛線部份。因此第二級的防禦方法，主要著眼於阻止這些後續的連線，由於些這後續的連線，缺乏明顯的特徵可供判斷，所以我們必須採用「Dynamic TCP Reset」的回應方法，以上述這個假設情況為例，其 Dynamic TCP Reset 將設定為：

> 當觀察到 X 透過 IIS Indexing Service Overflow 攻擊 Y 時，立即中斷 X 與 Y 間所有現有連線，並在 *x* 分鐘內，阻止 X 與 Y 建立任何其他連線。

在上述規則中，X、Y 的值必須等到入侵偵測系統觀察到「IIS Indexing Service Overflow 攻擊」後才能確定，所以稱之為 Dynamic TCP Reset Rule。

當 A 網段的 local sensor 採用第二級的防禦方法後，會先企圖中斷掉 A 與 B、C 的任何連線，而其他網段的 monitor 在收到 A 的緊告訊息後，會將原本的 dynamic rule 進行區域化的處理後，再命令旗下的 local sensor 採用，以預防網蟲的進攻。進行區域化的目的，主要是為了適應個網段不同的狀況，以最佳化回應機制或避免誤殺，例如：B 網域下可能有一台重要的 Server，原本就會有許多網路連線進進出出，為了避免該 dynamic TCP Reset rule 影響到該 Server 的正常運作，B 網域的 monitor 可能會將原本的 dynamic rule localize 改成：

> **If Y!=*Server***
> **then** 當觀察到 X 透過 IIS Indexing Service Overflow 攻擊 Y 時，立即中斷 X 與 Y 間所有現有連線，並在 *x* 分鐘內，阻止 X 與 Y 建立

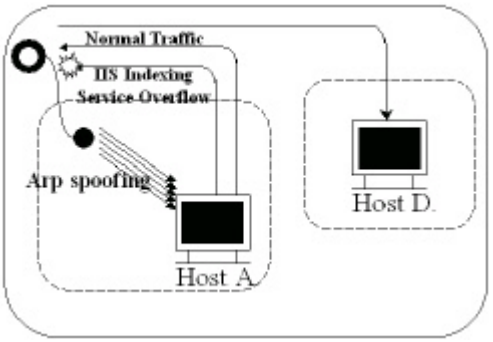
任何其他連線。

local sensor 在採用了第二級防禦措施：Dynamic TCP Reset 後，仍會繼續追蹤該回應方法的狀況與成效，例如各 local sensor 會向 monitor 報告，在設定該 dynamic rule 後，企圖中斷 Y 連線到 X 的 80 port 七次。當 monitor 持續收到這類統計數據後，就可判斷該網蟲會透過 ***http*** 進行後續的溝通，所以可將原本的 Dynamic rule 進行更進一步的精緻化，以改善回應效率且降低誤殺正常連線的可能。

若是該網蟲並沒有建立後續連線的行為(例如第一類網蟲)，或是我們的 local sensor 發現第二級的防禦措施仍無法有效防止網蟲的擴散(如觀察到 B、C 繼續利用 IIS Indexing Service Overflow，向外入侵 D、E、F 等)，各 local sensor 將會再度提昇危險等級，並開始採取更激烈的回應手段，並通報其他網段的 monitor，建議他們調整回應策略。

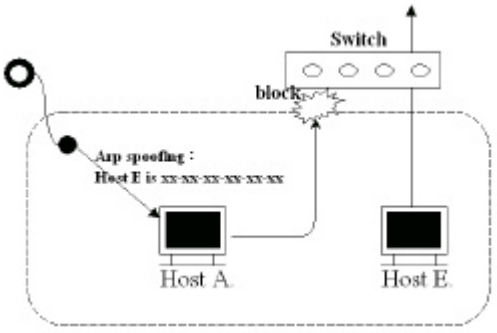
不同於前兩階段的 TCP Reset 方法中，local sensor 是站在第三者的立場企圖阻止攻擊的發生。在第三級的回應機制中，local sensor 主要是利用 arp spoofing 的方式主動將攻擊流量強制導引至 monitor 處，直接介入攻擊者與受害者的網路連線中，如圖六所示。其基本運作方式是透過 local sensor 主動以 arp reply 送給攻擊端主機，欺騙它該網域下所有主機的實體位址為 monitor 的 MAC，當攻擊端主機企圖與受害者端主機建立連線時，就會將封包送往 monitor，monitor 就可以針對這些封包內容進行檢查，若發現為攻擊封包則予以丟棄。由於 monitor 是站在攻擊者與受害者中間，所以幾乎能完全阻止攻擊行為。

不過這種方式的缺點在於 monitor 必須負責轉送所有由攻擊者主機的網路封包，而且 local arp 必須不斷送出 arp reply 強制更新攻擊者主機的 arp table，當網路中受網蟲感染的主機增加時，不但 monitor 可能難以承受這麼大的網路流量，大量的 arp reply 封包也可能成為網路極為沉重的負擔。所以各 control domain 的 monitor 若發現該網路受感染的主機實在太多了，必須繼續將危險等級提昇到第四級，以採取更有效的回應機制。



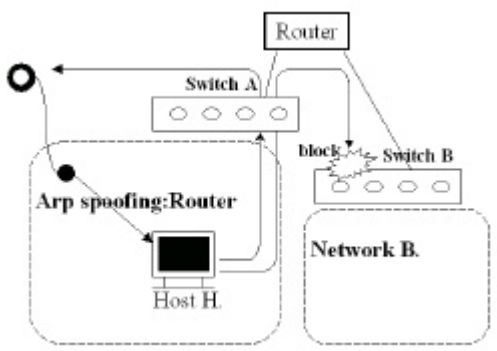
圖六：第三級的回應措施

第四級的回應機制主要是透過 snmp 重新設定 switch 的過濾規則，直接將攻擊者電腦的網路卡號鎖住(block)，這麼做等於完全阻絕了攻擊者電腦對外的網路通道，因此 monitor 可以不用再負責轉送攻擊者電腦的封包，local sensor 所需 spoofing 的範圍也可以縮小到只 spoofing 同一 switch 下的主機(而非整個區域網路)，如圖七所示。



圖七、第四級的回應方法

當然各個 monitor 也可以針對其管轄網域的個別狀況，進行特別特殊的處理。如圖八，由於主機 H 是該網路中非常重要的 Server，所以即使進入了第四級的危險等級，仍然要讓它能通過 switch A，將封包送至 monitor 處檢查，只暫時停止他對於 switch B 下機器的服務，透過這種方式好在防治網蟲擴散與維持網路服務正常運作間做出權衡。



圖八：具彈性的回應策略。

4.3 防治來自網際網路的網蟲攻擊

上面所介紹的方法主要是要防止網蟲由區域網路內的電腦向外擴散和避免網蟲於區域網路內交互感染，對於由網際網路向區域網路入侵的網蟲，我們則必須採取不同的防治措施。於不同危險等級將採用的防禦方法整理如下表：

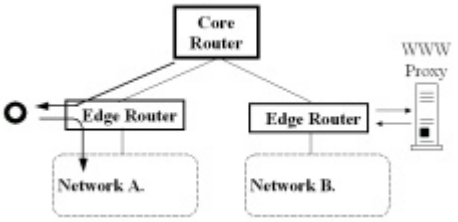
表四：防治網蟲由網際網路入侵

危險等級	可採取的回應措施
第一級	TCP Reset
第二級	Dynamic TCP Reset
第三級	IP Filter
第四級	Traffic Redirect

第一級、第二級的防護方法和之前提到的區域網路內的防護方法類似，也是希望利用最基本的 TCP Reset 就能檔掉的網蟲攻擊。當發現這種防治措施無效，且區域網路內其他的 monitor 已經發現有網蟲在活動的蹤跡時，負責控管 router 的 monitor 就會將危險等級提昇為第三級，並採取新的回應方法。

第三級的回應方法主要希望透過重新設定 router 的過濾規則，將不斷發出攻擊的 IP 位址鎖住，使該主機無法繼續對內部發動攻擊。不過這種回應機制的缺點在於，必須先觀察到攻擊行為，才能鎖住該 IP 位址，在大規模的網蟲攻擊事件中，通常會有成千上萬不同的主機，從網際網路對內部機器發動攻擊，這種被動的防守方式顯然無法阻止網蟲的攻擊。所以當 monitor 發現，有太多不同來源的攻擊主機時，就會緊急提昇危險等級至第四級，並開始採取新的防禦措施。

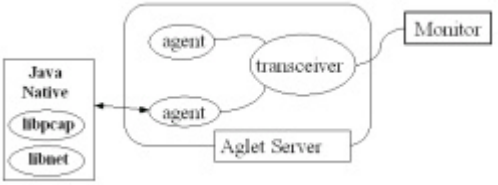
第四級的防禦方法主要將採用分散式的防禦措施，將防守的責任交給子網路的路由器和 monitor 處理。monitor 會重新設定子網路 router 的路由規則，將所有目的位址為該子網路的封包全部轉向 monitor 處理。若該 router 有支援 layer-4 的路由功能，則可僅針對網蟲所使用的通訊協定，導向相對應的 Application Proxy，例如在圖九中網蟲若是利用 http 協定在進行攻擊 Network B，則可僅將所有的 http 連線導向 www proxy 進行處理，由 www proxy 負責所有服務請求和過路攻擊行為。



圖九：第四級的回應機制。

五、系統實作

本分散式入侵偵測與回應系統的雛形已在建構中，其整體架構如圖十所示：



圖十：軟體架構圖

在 transceiver 與 agent 方面主要利用 IBM Aglet[2] API 來建置，不論是 transceiver 或 agent 都包裝成 aglet agent 的形式，不過目前所有 agent 都是 Stationary Agent，尚未實作 mobile agent。此外，許多 agent 會用到的 C Library，如：libpcap、libnet、libnids 都透過 Java Native 包裝成 Java Classes。在 monitor 方

面，則利用 Borland C++ Builder 5.0 來開發。另外我們還整合了 Snort IDS 成為我們重要的事件蒐集器。

在此系統中，主要的 agnet 包括了，SnortLogPaser Agent、SessionKeep Agent、Arp_Spoof Agent、SNMP_Set Agent，其主要功能整理如下表：

表六：主要 agent 的功能

名稱	主要功能
SnortLogPaser	讀取 Snort IDS 的 Alert Log，並重整格式，報告給 transceiver
SessionKeep	維護網路上每個 TCP Session 的狀態，並接收指示中斷特定網路連線
Arp_Spoof	負責製造 arp spoof 的封包
SNMP_Set	利用 snmp 模組，設定 Switch 上的過濾規則

接下來我們將簡略說明上一節所提到的各種回應機制是如何利用這些 agents、transceivers、monitor 實作出來。

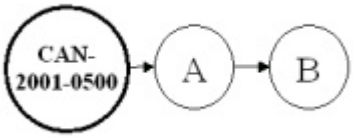
(1)TCP Reset 機制的實作

TCP Reset 的實作主要透過 Snort IDS 的 flexible response 來設定。如下面的設定規則：

```
alert tcp any any => 192.168.1.0/24 80 (resp: rst_all;msg: “IIS Unicode Attack”; content: “/scripts/..%35c../winnt/system32/cmd.exe?”)
```

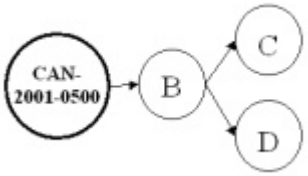
(2)判斷網路中是否有網蟲

Monitor 必須評估回應機制是否有效，主要的衡量標準是觀察網蟲是否持續擴散，我們將此功能實作於 monitor 的 graph engine 上。我們的 Graph Engine 主要改寫自劉順德[1]所開發的系統，其運作過程如下，Monitor 會持續接收來自於 local sensor 的入侵事件報告，並將入侵事件報告以樹狀結構表示。例如：當我們觀察到電腦主機 A 利用 IIS Indexing Services Buffer Overflow 攻擊電腦主機 B 時，graph engine 就會產生一個 MetaNode，且將其 ID 設為 CAN-2001-0500(Indexing Service ISAPI Extension Buffer Overflow Vulnerability 的 CVE 編號)，然後為參與此事件的 A、B 主機各別產生兩個子節點，並利用有向邊將它們和 MetaNode 連結起來，以表示 A 用 CAN-2001-0500 的漏洞攻擊 B，如圖十一所示。



圖十一：以樹狀表示法事件

若 monitor 觀察到 B 利用相同的漏洞來攻擊 C、D 時，也會用相同的表示法表示，如圖十二所示。



圖十二：以樹狀表示法事件

當 Graph Engine 發現有兩顆樹，它們的 MetaNode ID 都為 CAN-2001-0500 時，就會將兩顆樹加以合併，若合併後樹的深度超過警界值，就表示 Indexing Services Buffer Overflow 攻擊已經擴散，極有可能是網蟲正在活動。

圖十三：將兩樹狀結構加以關聯

(3)Dynamic TCP Reset 機制的實作

在研究了各種現有軟體後，我們發現並沒有產品能完全滿足 dynamic TCP Reset Rules 的需求，其中與我們設計較接近的是 snort 所提供的 active/dynamic rules，不過由於其 dynamic rules 仍無法動態決定要處理的 IP 位址，所以仍無法直接套用。

所以在這部份我們採取較間接的方式來實作 dynamic TCP Reset rules，我們利用 KeepSession Agent 來負責維護網路上 TCP Session 的狀況。當 Snort 偵測到入侵行為時，會產生 alert log，由 SnortLog_Parse 讀取了後報告給 transceiver。若此時已經進入第二級危險等級，transceiver 則會繼續將此訊息轉送給 KeepSession Agent，由 KeepSession Agent 負責將相關 TCP Session 中斷。

KeepSession Agent 主要利用 libpcap、libnet 兩 classes 來實作監聽封包與中斷 Session 等功能。

(4)Arp-Spoofing 機制的實作

我們的 Arp-Spoof Agent 主要改寫自 dsniff[11]中的 **arpspoof.c**。比較不同的是，我們的 Arp-Spoof Agent 是將所有區域網路內電腦 Spoof 成第三者 monitor 的卡號。

(5)Monitor 檢查與轉送網路封包的實作方法

雖然 monitor 檢查與轉送網路封包的動作，非常類似一般 router 或 firewall 的工作，不過由於我們尚未找到容易寫程式和容易與我們系統整合的產品，所以我們並沒有利用現成的 router 或 Firewall 來實作。

目前我們是利用 libpcap 來收取封包，經檢查後(目前僅對個別封包的內容進行檢查，並未將進行封包重組)，再利用 libnet 發送到真正的目的地。

這種做法的缺點在於，由於目前 libnet 僅能支援一張網路卡，收與發的動作都必須透過同一張網路卡，所以效率會受到影響。這是日後亟待改進的地方。

(6)Router 動態設定的實作

在此雛形系統中，我們的 router 主要是使用 FreeBSD 來架設，所以可以容易地透過簡單的 shells scripts 來變更路由設定。此外我們還啟動 FreeBSD 上的 IP Firewall，並利用 ipfw(controlling utility for IP firewall and traffic shaper)來設定要阻擋的 IP 位址。

六、結論

CodeRed 網蟲的出現，已經徹底改變了網路入侵的型態，網蟲已成為全球網路最嚴重的安全威脅。有鑑於此，本文特別針對如何偵測與防治網蟲進行研究。

文中我們提出了一套分散式入侵偵測與回應系統。這套系統主要特色包括了：(1)整合了 switch、router 等網路設備，以增強反制能力，(2)利用 graph engine 來進行關聯分析，以提昇對於網蟲的認知能力，(3)採用動態的回應機制，以加快回應速度，及對抗各不同種類型的網蟲。

論文中我們分別從：(i)如何防治網蟲在區域網路內或對外的攻擊，和(ii)如何防治網蟲由網際網路向內攻擊兩個角度來討論此分散式架構的應用。最後我們也說明實作的方式。

在未來我們除了繼續完成系統實作與改善系統效能外，還希望能將這套分散式防護系統實際應用在真實的網路環境中，以確實發揮抵禦網蟲入侵的功能。

七、參考文獻

[1] 劉順德, "以樹狀關聯式架構偵測電子郵件病毒之探討", 中央大學資管系碩士論文, 2001 年 6 月.

[2] Aglets.org, "The aglets portal", http://aglets.sourceforge.net, 2001.

[3] CAIDA, "CAIDA Analysis of Code-Red", http://www.caida.org/analysis/security/code-red/, August 2001.

[4] CERT Coordination Center, "CERT Incident Note IN-2001-01: Compromises via ramen Toolkit", http://www.cert.org/ incident_notes/IN-2001-01.html, January 18, 2001.

[5] CERT Coordination Center, "CERT Advisory CA-2001-11 sadmind/IIS Worm", http://www.cert.org/advisories/CA-2001-11.html, May 08, 2001.

[6] CERT Coordination Center, "CERT Advisory CA-2001-19 Code Red Worm Exploiting Buffer Overflow In IIS Indexing Service DLL", http://www.cert.org/advisories/CA-2001-19.html, July 19, 2001.

[7] CERT Coordination Center, "CERT Advisory CA-2001-26 Nimda Worm", http://www.cert.org/advisories/CA-2001-26.html, September 25, 2001.

[8] CurtisA. Carver, Jr., John M.D Hill, John R. Surdu Member, IEEE, and Udo W. Pooch, Senior Member, IEEE, "A Methodology for Using Intelligent Agents to provide Automated Intrusion Response", Proceedings of the 2000 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 6-7 June, 2000.

[9] D. Schnackenberg, K. Djahandari, and D. Steme, "Infrastructure of Intrusion Detection and Response", Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, SC, January 2000.

[10] Dan Schnackenberg, Harley Holliday, Randall Smith, Kelly Djanhandari and Dan Steme, "Cooperative Intrusion Traceback and Response Architecture", DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings, Vol: 1, Page(s): 56-68, 2001.

[11] Dug Song, "dsniff", http://www.monkey.org/~dugsong/dsniff/

[12] Guy Helmer, Johnny S. K. Wong, Vasant Honavar, and Les Miller, "Lightweight Agents For Intrusion Detection", Submitted to Journal of Systems and Software, http://latte.cs.iastate.edu/~ghelmer/NewFac.ps, November 27, 2000.

[13] Guy Helmer, et al, "Software Fault Tree and Colored Petri Net Based Specification Design and Implementation of Agent-Based Intrusion Detection Systems", Submitted to ACM TISSEC, http://latte.cs.iastate.edu/~ghelmer/CPNIDS.ps, January 18, 2001.

[14] Jai Sundar Balasubramaniyan, Jos Omar Garcia-Fernandez, David Isacoff, Eugene Spafford, Diego Zamboni, "An Architecture for Intrusion Detection Using Autonomous Agents", Technical Report 98/05, COAST Group, Purdue University, 1998.

[15] Jose Nazario, Jeremy Anderson, Rick Wash and Chris Connelly, "The Future of Internet Worms", The Black Hat Briefings '01 July 11-12th Las Vegas , July 20, 2001.

[16] Joseph Barrus, Neil C. Rowe, "A Distributed Autonomous-Agent Network-Intrusion Detection and Response System", the Proceedings of the 1998 Command and Control Research and Technology Symposium, Monterey CA, 1998.

[17] Lee Gabaer, "Melissa Virus Create a New Type of Thread", IEEE Technology News: Computer, Vol. 32, No. 6, Page(s): 16-19, June 1999.

[18] Mark Crosbi, Gene Spafford, "Active Defense of a Computer System Using Autonomous Agents", Technical Report No. 95-008, COAST Group, Purdue University, 1995.

[19] Mark Slagell, "The Design and Implementation of MAIDS", A creative component submitted to the graduate faculty in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE, May 10, 2001.

[20] Mike Petkac and Lee Badger, "Security Agility in Response to Intrusion Detection, Computer Security Applications", 2000. ACSAC '00. 16th Annual Conference, Page(s): 11-20, 2000.

[21] Phillip A. Porras and Peter G. Neumann., "EMERALD: Event monitoring enabling responses to anomalous live disturbances", Proceedings of the 20th National Information Systems Security Conference, National Institute of Standards and Technology, 1997.

[22] SANS Institute, "SANS Global Incident Analysis Center > Lion Worm", http://www.sans.org/y2k/lion_protection.htm, March 26, 2001.

[23] Snort.org, "Snort - The Open Source Network IDS", http://www.snort.org

[24] Steven Cheung, et al, "The Design of GrIDS: a graph based intrusion detection system", Department of Computer Science, University of California at Davis, http://seclab.cs.ucdavis.edu/arpa/grids/grids.ps, January 1999.