植基於 Flow 資訊之網頁為基礎之殭屍網路偵測

Web-Based Botnet Detection Based on Flow Information

蔡育洲
中山大學資管系
j840705j@hotmail.com

林孝忠
崑山科技大學資管系
fordlin@mail.ksu.edu.tw

陳嘉玫
中山大學資管系
cchen@mail.nsysu.edu.tw

***摘要*—殭屍網路(Botnet)為目前諸多網路攻擊之主要來源。攻擊者(Botmaster)經由網際網路控制受感染之電腦，並經由 IRC(Internet Relay Chat)、P2P(Peer-to-Peer)、Web(HTTP)等不同方式下達控制與命令指令。受感染之電腦接收指令後，可進行諸如 DDoS(Distributed Denial-of-Service) 、發送垃圾郵件(SPAM)等攻擊。Web-based Botnet 經由標準 HTTP 協定進行控制，將自身流量隱藏在正常網頁流量下，因此更加難以判斷與偵測。本研究主要著重在 Web-based Botnet 之分析與偵測，並以 Web-based Bot 本身的特性進行偵測。同時亦配合多種實驗環境設計進行偵測實驗，均得到良好的偵測結果。此外，在真實網路環境下進行實驗亦能達到高偵測率。**

Abstract—Botnet is the major source of Cyber attacks. Botmaster controls and commands the infected hosts to launch DDoS (Distributed Denial-of-Service) attack or send SPAM through some public protocols such as IRC (Internet Relay Chat), P2P (Peer-to-Peer) or Web (HTTP). Web-based Botnet is much difficult to detect its existence because of the command and control messages of Web-based Botnet issued through HTTP protocol and hidden behind normal web network traffic.

In this research, we focus on analysis and detection of Web-based Botnet according to the features of Web-based Bots. The experiment shows the proposed approach gets good results in many different topology designs. It also shows high detection rate in real network environment.

***關鍵詞*—僵屍網路、僵屍程式、網頁為基礎僵屍網路**

***Keyword*—Botnet、Bot、Web-based Botnet**

一、前言

僵屍網路(Botnet)目前已成為一個攻擊平台，攻擊者(Botmaster)利用該平台可以有效地發起各式各樣之攻擊行為，導致整個基礎資訊網路或重要應用系統癱瘓，竊取大量機密資料或洩漏個人隱私，以及從事網路欺詐活動，甚至可被用來發起新的未知攻擊。殭屍網路指採用一種或多種傳播手段，將大量主機感染 Bot 程式(殭屍程式)，從而在控制者(Botmaster)與被感染主機間所形成一個可一對多之控制網路。透過僵屍網路發動之攻擊行為，描述如下所示：

(1).拒絕服務攻擊(Distributed Denial-of-Service, DDos)：攻擊者透過所控制之 Bots，使其在特定的時間同時開始連續訪問特定的網路目標，從而達到 DDoS 之目的。

(2).發送垃圾郵件(SPAM)：Bots 可透過 Socks v4 或 Socks v5 代理(Proxy)功能，發送大量垃圾郵件，而且發送者可以很好地隱藏自身之 IP 資訊。

(3).竊取機密資料：僵屍網路之控制者可以從受感染之主機竊取使用者之各種敏感資訊與其他隱私資料，諸如個人銀行帳號與密碼或信用卡號碼等。同時 Bots 能夠使用探測器(Sniffer)觀查與蒐集感興趣之網路資料，從而獲得網路流量中之私密資訊。

(4).濫用資源：攻擊者透過僵屍網路從事各種耗費網路資源之活動，從而令使用者之網路性能受到影響，甚至帶來經濟損失，諸如植入廣告軟體，或利用受感染之主機搭建假冒之銀行網站從事網路釣魚(Phishing)非法活動。

僵屍網路控制方式從透過 IRC(Internet Relay Chat)作為溝通媒介，進而藉由 Web 控制 Botnet。Web-based Botnet 經由標準 HTTP(Hyper Text Transfer Protocol)進行控制，將自身流量隱藏在正常網頁流量下，因此難以判斷與偵測。Web-based Botnet 具有以下描述之特性，使其偵測不易：

(1).流量不大：Bot 在感染前，透過 Port Scan 探索未受害電腦主機之弱點，進而針對弱點進行感染動作。除非 Bot 掃描大量 IP，否則 Port Scan 所產生之流量並不大。感染後之電腦主機，除了與 Command and Control (C&C) 伺服器(Server)溝通外，平時不會有其他網路行為，因此潛伏在受害電腦主機中，不容易發覺其蹤跡。

(2).符合正常協定：Bots 所使用之 Port 可能為正常的 80 Port，或其他常使用之 Port Number，因此容易隱藏在正常流量下，不易偵測。

(3).可觀察之感染數量不定：在平常之學術網路或其他網路環境中，無法瞭解該網路之某種 Bot 之數量多寡。

有鑑於上述原因，必須思考如何在一封閉環境下觀察 Bots、C&C 伺服器、與攻擊者間之關係，並模擬 Web-based Botnet 之形成各階段，以觀察其流量變化，並挖掘與分析 Web-based Botnet 之特有特徵，以偵測 Web-based Botnet。本研究主要針對 Web-based Botnet 進行研究及實驗，以期發展出一 Web-based Botnet 偵測系統。

本研究主要為發展出以 NetFlow [2] 為基礎之 Web-based Botnet 偵測方法，並利用 Web-based Botnet 之特性，諸如時間連接特性(Timeslot)，以及 NetFlow 提供之資訊，作為偵測特徵。主要針對 NetFlow 流量進行檢測，並判斷是否為 Web-based Bot，以提供網路管理人員快速偵測 Web-based Botnet 之架構，並能進一步處理受感染之電腦或採取相關應對措施。

本研究之研究目的主要為(1)進行 Web-based Botnet 實驗，挖掘與分析 Web-based Botnet 之特性，(2)提出一個 Web-based Botnet 偵測架構，並以 NetFlow 分析 Web-based Botnet 特有之特性，再依特性區別正常流量與異常流量之異同，以及(3)實作 Web-based Botnet 偵測系統雛型，同時在封閉環境以及實際網路環境進行實作與偵測，以瞭解系統偵測之準確性與誤判率。

二、文獻探討

有關僵屍網路研究方面，Cooke et al. [3] 提出關於僵屍網路的 Command and Control (C&C) Channel 可分為三大類，其一為集中式(Centralized)，例如攻擊者透過 IRC Server 下達攻擊命令，此種模式的僵屍網路較容易偵測與防治，攻擊者也容易控制其 Bots 與傳遞訊息。其二為 P2P(Peer-to-Peer)，此種模式的僵屍網路破壞其中一個 Bot，不代表已經破壞整個網路架構，也提供攻擊者一定程度的可控制性。最後則為 Random Model，主要透過隨機掃瞄網路上主機之漏洞以進行感染，使其成為 Bot，當遇到受樣同樣 Bot 感染的主機時，彼此互通訊息以建立 Botnet，由於訊息傳遞延遲性高，也不保證可以正確送達 Bots，對攻擊者說，其可控制性較差，建立僵屍網路的複雜度提高，不容易為攻擊者使用。

Rajab et al. [12]利用 Honeynet 技術，針對 IRC-Based Botnet 進行長達三個月以上的追蹤，分析其成長與擴散模式，以及生命週期。其研究結果顯示，僵屍網路的網路規模從好幾百至上千個都有，同時產生大量的惡意之網路連結流量，並且產生向 DNS Server 查詢特定 DNS 網域名稱的證據。

Frelling et al. [4] 利用 Honeypot 技術針對利用 IRC-Based Botnet 進行 DDoS 攻擊觀測其攻擊行為，並利用其開發之軟體，分析其捕捉之 Bots，以便對於 IRC Based Botnet 之 DDoS 攻擊

有更深入之認識。

Barford and Yegneswaran [1] 藉由分析 Agobot、SDBot、SpyBot、以及 GT Bot 的原始碼，將其功能分門別類說明，使得 IRC-Based Botnet 之特性與網路架構更加清楚。其研究結果對於瞭解 IRC-Based Botnet 貢獻頗大。由於 Agobot 具有模組化功能，可供攻擊者隨意擴充其功能，衍生許多變形版本，增加偵測難度。

Strayer et al. [9] [14] 使用一些機器學習(Machine Learning)的技術，諸如 C4.5 Decision Tree、Naïve Bayes、Bayesian Networks，萃取相關的網路流量屬性，諸如 Duration、Bytes-Per-Packet、Bits-Per-Second 等，以區別正常的 IRC 流量與 IRC-Based Botnet 的流量。隨後利用相關分析(Correlation Analysis)找出 Flows 間的關係，以確定是否來自同一 Bot。

Karasaridis et al. [7] 利用傳輸層資料分析僵屍網路，透過建立正常流量與異常流量模式，並比較其間的差異，以產生警告訊息。根據所產生的警告訊息，利用相關分析，找出來自同一 Bot 之流量。接著，結合分析來自應用層的資料，以降低誤報率。

Gu et al. [5] 利用入侵偵測系統產生的警告訊息，進行相關分析，以判斷是否有惡意軟體，諸如 Bot 之入侵行為發生，並實作出 BotHunter 系統。此 BotHunter 系統建基於 Snort [13] 入侵偵測系統，主要實作兩個演算法，分別為 SLADE(Statistical payload Anomaly Detection Engine)，以及 SCADE(Statistical sCan Anomaly Detection Engine)，以 Plug-in 的方式增強 Snort 之功能。接著，結合 Snort 的規則庫(Rule Base)，以判斷有無 Bot 活動或入侵事件發生。

由於 BotHunter 利用入侵偵測系統產生之警示訊息偵測僵屍網路，可能無法有效偵測出新型 Bot 行為。隨後 Gu et al. [6] 針對 IRC-based 以及 Web-based Botnet 提出 BotSniffer 分析方法。此分析引擎不僅分析 IRC 訊息，同時也針對 Bot 所進行的活動，諸如掃瞄其他主機、發送垃圾郵件等活動進行分析，並透過相關分析以判斷僵屍網路的存在與否。

Team Cymru [16] 與 Lee et al. [8]的研究主要著重於 Web-based Botnet。Team Cymru 將目前 Web-based Botnet 之分布與數量等資訊進行統計分析，並說明 Web-based Botnet 之嚴重性。Lee et al.發現 Blackenergy 具有時間特性。

由以上的文獻可知，目前有關僵屍網路的研究主要集中於 IRC-Based Botnet。除了針對所管轄的網路進行流量分析外，隨後也針對 Web-based Botnet 提出偵測方式。

三、Web-based Botnet-Blackenergy 分析

本研究以 Blackenergy [8]為基礎，挖掘與分析 Web-based Botnet 之特性。其主要運作方式為，攻擊者將指令存放在某台 Web Server 上，以進行命令與控制 Bots 之用。Bots 再定時與 Web Server 連線取得指令，以決定攻擊目標與參數設定。Web-based Botnet 連線架構如圖 1 所示：

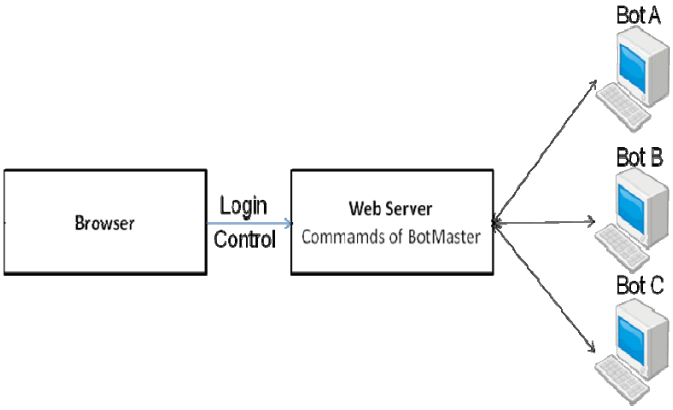


圖 1. Web-based Botnet 連線架構

由圖 1 可知，攻擊者透過瀏覽器開啟 Web Server 上的操作頁面，並將指令存放在 Web Server 上之資料庫中，Bots 則定時至 Web Server 上取得並執行指令。

為找出 Blackenergy 之特性，將透過 NetFlow 流量與 Payload 分析，藉此兩種不同分析來源驗證 Blackenergy 所具有之特性，以瞭解其運作時流量與正常流量之異同處。圖 2 為利用成功大學 Testbed [17] 所架設之 Blackenergy 實驗環境，以 Router 串連 Server 端、Bot 端、以及受害端等電腦主機，進行攻擊實驗，達成模擬真實環境之目的。

以下依序描述實驗環境中各角色之功能：

(1).Server 端：此 Server 內安裝 Apache、PHP、以及 MySQL，放置 C&C 操控網頁以供攻擊者操控 Bots 之用，而 Bot 依參數之設定定時與 Server 連線取得資料。

(2).Bot 端：已事先安裝 Blacknergy，其設定每一分鐘與 Server 連線一次。每次連線 Bot 會向 MySQL 取得最新指令、攻擊目標，或更新 Bot 功能之連結網址。

(3).BotMaster：BotMaster 不管在何處，只要能連上網路，透過瀏覽器(Broswer)開啟網頁即可連線到 Server 下達攻擊指令。Botmaster 所下達之所有指令與參數皆會紀錄在 MySQL 資料庫中，等待 Bot 取得資訊。

(4).Router 端：模擬串連 Bot 與 Victim 之設備，所有的封包與流量皆會經過此 Router，在此 Router 上以 Tcpdump [15] 收集所有進出之流量與封包資訊。

(5).Victim：為 Bot 攻擊之對象。

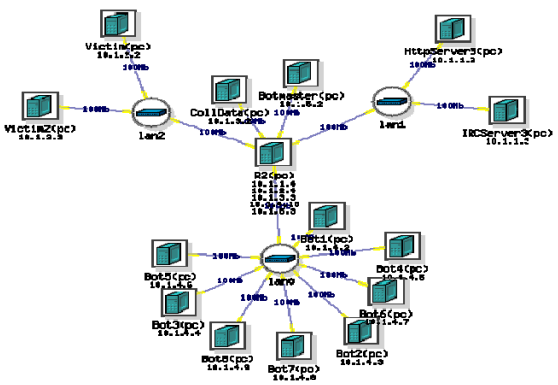


圖 2. Blackenergy 實驗拓樸

在流量收集部份，主要收集兩種資料，一為「.pcap」資料，另一為 NetFlow 資料。在 Blackenergy 攻擊之前，已事先部署資料收集主機(CollData)以收集「.pcap」資料，並以 Wireshark [18] 分析其內容。同時設定 Router 將 Netflow V5 格式導出並傳送至 CollData 主機，以收集 NetFlow 資料，並以 NFDUMP [10] 搭配 NFSen [11] 分析 NetFlow 資料。

Blackenergy 運作情形觀察重點如下所述:

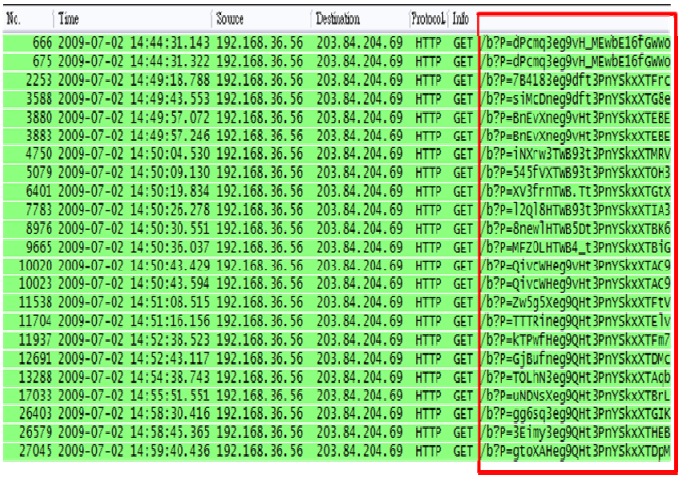
(1).Botmaster 與 Server 間之流量：由於 Blackenergy 採用標準 HTTP 進行連線與傳輸，因此可以針對 Port 80 進行監控，期能從 Botmaster 下達指令後之流量變化尋找 Blackenergy 活動之蛛絲馬跡。

(2).Server 與 Bots 間之流量：Bots 與 Server 間連線有個重要的時間特性，即 Bot 會依照參數之設定時間與 Server 連線，因此產生下列現象，一為 Bots 定時與 Server 連線(端看參數設定之時間，最短為一分鐘)，另一為 Bots 每次與 Server 連線後，取得之資料量非常近似。Bots 所查詢之資料為 MySQL 資料庫 內「opt」資料表(Table)之 13 個數值，如圖 3 所示。

name	value
attack_mode	0
cmd	flood icmp 140.117.241.197
http_freq	50
http_threads	5
icmp_freq	5000
icmp_size	5000
max_sessions	30
spoof_ip	0
syn_freq	10
tcpudp_freq	500
tcp_size	2000
udp_size	10000
ufreq	1

圖 3. Blackenergy 參數設定

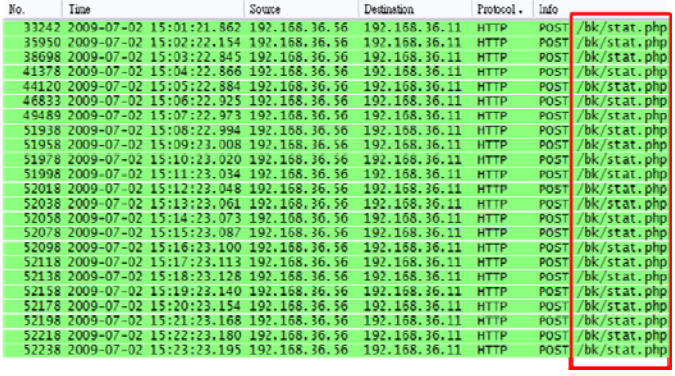
首先，從成功大學 Testbed 模擬實驗環境其中一台電腦之封包 Payload，其包含正常與異常流量，分析 Blackenergy 之連線特性。圖 4 顯示 IP 192.168.36.56 與 IP 203.84.204.69 (Yahoo Server) 之連線，其 Time 欄位無呈現明顯特性，紅線所框示之 Info 欄位可看出每次與 Yahoo Server 連線之頁面皆不相同。由以上描述可知，IP 192.168.36.56 每次與同一個 IP 203.84.204.69 連線，其所 Request 之頁面皆不同。



No.	Time	Source	Destination	Protocol	Info
666	2009-07-02 14:44:31.143	192.168.36.56	203.84.204.69	HTTP	GET /b?P=dPcmq3eg9vH_MEwbE16fGWW0
675	2009-07-02 14:44:31.322	192.168.36.56	203.84.204.69	HTTP	GET /b?P=dPcmq3eg9vH_MEwbE16fGWW0
2253	2009-07-02 14:49:18.788	192.168.36.56	203.84.204.69	HTTP	GET /b?P=7B4183eg9dft3PnYSkxXTFrc
3588	2009-07-02 14:49:43.553	192.168.36.56	203.84.204.69	HTTP	GET /b?P=siMcDneg9dft3PnYSkxXTG8e
3880	2009-07-02 14:49:57.072	192.168.36.56	203.84.204.69	HTTP	GET /b?P=8nEvXneg9vHt3PnYSkxXTEBE
3883	2009-07-02 14:49:57.246	192.168.36.56	203.84.204.69	HTTP	GET /b?P=8nEvXneg9vHt3PnYSkxXTEBE
4750	2009-07-02 14:50:04.530	192.168.36.56	203.84.204.69	HTTP	GET /b?P=1NXrw3TwB93t3PnYSkxXTMRV
5079	2009-07-02 14:50:09.130	192.168.36.56	203.84.204.69	HTTP	GET /b?P=545fVXTwB93t3PnYSkxXTOH3
6401	2009-07-02 14:50:19.834	192.168.36.56	203.84.204.69	HTTP	GET /b?P=xV3frnTwB.Tt3PnYSkxXTGtX
7783	2009-07-02 14:50:26.278	192.168.36.56	203.84.204.69	HTTP	GET /b?P=l2Ql8HTwB93t3PnYSkxXTIA3
8976	2009-07-02 14:50:30.551	192.168.36.56	203.84.204.69	HTTP	GET /b?P=8nEwlHTwB5Dt3PnYSkxXTBK6
9665	2009-07-02 14:50:36.037	192.168.36.56	203.84.204.69	HTTP	GET /b?P=MFZOLHTWB4_t3PnYSkxXTBiG
10020	2009-07-02 14:50:43.429	192.168.36.56	203.84.204.69	HTTP	GET /b?P=QivcWHeg9vHt3PnYSkxXTAC9
10023	2009-07-02 14:50:43.594	192.168.36.56	203.84.204.69	HTTP	GET /b?P=QivcWHeg9vHt3PnYSkxXTAC9
11538	2009-07-02 14:51:08.515	192.168.36.56	203.84.204.69	HTTP	GET /b?P=Zw5g5Xeg9QHt3PnYSkxXTFtV
11704	2009-07-02 14:51:16.156	192.168.36.56	203.84.204.69	HTTP	GET /b?P=TTTRineg9QHt3PnYSkxXTElv
11937	2009-07-02 14:52:38.523	192.168.36.56	203.84.204.69	HTTP	GET /b?P=kTPwfHeg9QHt3PnYSkxXTFm7
12691	2009-07-02 14:52:43.117	192.168.36.56	203.84.204.69	HTTP	GET /b?P=GjBufneg9QHt3PnYSkxXTDMc
13288	2009-07-02 14:54:38.743	192.168.36.56	203.84.204.69	HTTP	GET /b?P=TOLhN3eg9QHt3PnYSkxXTAcb
17033	2009-07-02 14:55:51.551	192.168.36.56	203.84.204.69	HTTP	GET /b?P=uNDVsXeg9QHt3PnYSkxXT8rL
26403	2009-07-02 14:58:30.416	192.168.36.56	203.84.204.69	HTTP	GET /b?P=gg6sq3eg9QHt3PnYSkxXTGIK
26579	2009-07-02 14:58:45.365	192.168.36.56	203.84.204.69	HTTP	GET /b?P=3EimY3eg9QHt3PnYSkxXTHEB
27045	2009-07-02 14:59:40.436	192.168.36.56	203.84.204.69	HTTP	GET /b?P=gtoXAHeg9QHt3PnYSkxXTDpM

圖 4. 正常流量之封包分析

圖 5 為 IP 192.168.36.56 與 IP 192.168.36.11 之連線數據。其中 192.168.36.11 為本研究所架設供 Bot 連線之 Web Server。此時已將 IP 192.168.36.56 感染成為 Bot。以 Time 欄位來看，可看出 IP 192.168.36.56 每隔約一分鐘向 IP 192.168.36.56 POST 一次資料，且紅線所框示之 Info 欄位顯示，每次皆 POST 同一個頁面「/bk/stat.PHP」，此為異常情形。



No.	Time	Source	Destination	Protocol	Info
33242	2009-07-02 15:01:21.862	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
35950	2009-07-02 15:02:22.154	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
38698	2009-07-02 15:03:22.845	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
41378	2009-07-02 15:04:22.866	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
44120	2009-07-02 15:05:22.884	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
46833	2009-07-02 15:06:22.925	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
49489	2009-07-02 15:07:22.973	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
51938	2009-07-02 15:08:22.994	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
51958	2009-07-02 15:09:23.008	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
51978	2009-07-02 15:10:23.020	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
51998	2009-07-02 15:11:23.034	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52018	2009-07-02 15:12:23.048	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52038	2009-07-02 15:13:23.061	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52058	2009-07-02 15:14:23.073	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52078	2009-07-02 15:15:23.087	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52098	2009-07-02 15:16:23.100	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52118	2009-07-02 15:17:23.113	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52138	2009-07-02 15:18:23.128	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52158	2009-07-02 15:19:23.140	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52178	2009-07-02 15:20:23.154	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52198	2009-07-02 15:21:23.168	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52218	2009-07-02 15:22:23.180	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php
52238	2009-07-02 15:23:23.195	192.168.36.56	192.168.36.11	HTTP	POST /bk/stat.php

圖 5. 異常流量之封包分析

接著，從 NetFlow 流量資訊觀察 Blackenergy 之連線情形。表 1 為片段摘錄資訊，”SrcIP”為來源主機之 IP，”DstIP”為目的主機之 IP，”Time Start”為 Flow 起始時間，”Time End”為 Flow 結束時間，”Interval”則為次一筆紀錄之時間與當筆紀錄之時間差。由表 1 可看出每次連線間隔時間為 180 秒，恰好與實驗所設定之連線時間相符。正常使用者瀏覽網頁之 Flow 資訊，不會有如此規律情形發生，因此可確認 Blackenergy 具有定時連線(Timeslot)特性。

表 1. Blackenergy 之時間特性

SrcIP	DstIP	Time Start	Time End	Interval
10.1.4.2	10.1.1.2	14:13:22.156	14:13:22.164	180
10.1.4.2	10.1.1.2	14:16:22.199	14:16:22.206	180
10.1.4.2	10.1.1.2	14:19:22.234	14:19:22.242	180
10.1.4.2	10.1.1.2	14:22:22.277	14:22:22.284	180
10.1.4.2	10.1.1.2	14:25:22.312	14:25:22.320	180
10.1.4.2	10.1.1.2	14:28:22.355	14:28:22.362	180
10.1.4.2	10.1.1.2	14:31:22.390	14:31:22.398	180

表 2 為 Blackenergy 之 NetFlow 欄位近似特性之片段摘錄資訊。

表 2. Blackenergy 之 NetFlow 欄位近似特性

SrcIP	SrcPort	DstIP	Packets	Bytes	Bpp
10.1.4.2	1343	10.1.1.2	5	490	98
10.1.4.2	1345	10.1.1.2	5	490	98
10.1.4.2	1372	10.1.1.2	5	490	98
10.1.4.2	1374	10.1.1.2	5	490	98
10.1.4.2	1396	10.1.1.2	5	490	98
10.1.4.2	1399	10.1.1.2	5	490	98
10.1.4.2	1413	10.1.1.2	5	490	98

由表 2 可知，每次 Bots 與 Server 間連線資料之 Packets、Bytes、Bpp(Bytes per packet)必相同或近似。

圖 6 以 Bpp 為例，將 Bots 之欄位資料以圖

形化方式呈現，橫軸為第 n 筆 Flow，縱軸為 Bpp 欄位數值。由圖 6 可看出五個遭受感染之電腦主機，其欄位數值隨時間持續保持非常近似之數值。由此可知，Bots 與 Web Server 間連線之 NetFlow 之值有非常近似的規律存在。

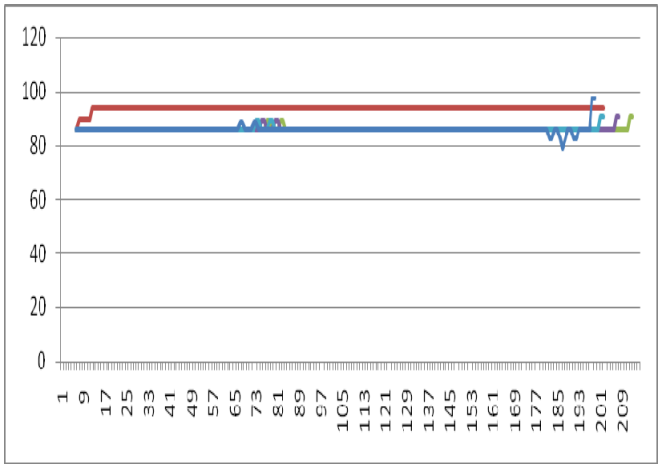


圖 6. Blackenergy 之 Bpp 流量

圖 6 之產生原因在於，每次連線內容相同，封包大小也相同，因此 Packets、Bytes、Bpp 等欄位資料可作為判斷是否為 Bots 之依據。在本實驗中，此特性之判斷主要為計算 NetFlow 每個欄位值之平均數與標準差。

經由本研究多次實驗 Blackenergy 之感染、控制、與攻擊過程，並收集封包分析，發現此 NetFlow 欄位之近似特性經過計算後，標準差約在 1 以下，而正常使用者之流量經過相同計算方式，並無如 Blackenergy 般極小標準差產生，因此 NetFlow 欄位之近似特性可用來偵測 Blackenergy。

在實驗過程中發現，Blackenergy 具有 Bot to Server(B2S) 與 Server to Bot(S2B) 之特性。B2S 指 Bot 至 Server 間所產生之特性，而 S2B 則指 Server 回傳資訊至 Bot 間產生之特性。表 3 與表 4 個別顯示部份正常流量與 Blackenergy 連線 NetFlow 之 B2S 與 S2B 特性。以表 3 之 Bpp 欄位資料來看，每次連線所產生的 Bpp 皆不同。第一次為 312，回傳值為 983;第二次為 316，回傳值則為 1151。而以表 4 異常流量之 Bpp 欄位資料來看，第一次為 98，回傳值為 103;第二次也為 98，回傳值為 103。

表 3. 正常流量之 NetFlow 分析

SrcIP	Src-Port	DstIP	Dst-Port	Bpp
203.70.206.232	4836	116.214.7.222	80	312
116.214.7.222	80	203.70.206.232	4836	983
203.70.206.232	5134	116.214.7.222	80	316
116.214.7.222	80	203.70.206.232	5134	1151

表 4. Blackenergy 之 NetFlow 分析

SrcIP	SrcPort	DstIP	DstPort	Bpp
10.1.4.2	1527	10.1.1.2	80	98
10.1.1.2	80	10.1.4.2	1527	103
10.1.4.2	1761	10.1.1.2	80	98
10.1.1.2	80	10.1.4.2	1761	103

造成異常流量具有 B2S 與 S2B 特性的主要原因在於，Blackenergy 取得指令之頁面皆為同一頁「/bk/stat.PHP」。在一段時間內若指令不改變，則取得之封包大小與 Timeslot 近乎相同。通常 Botmaster 為不產生多餘之動作以被偵測其存在，短時間內並會更改指令。正常使用者在同一台 Server 上所 Request 之頁面皆不相同，所得之封包大小也就不同。由上述說明可知，可利用 B2S 與 S2B 特性進行雙向驗證，為可行之偵測 Web-based Botnet 方法。

四、Web-based Botnet 偵測架構

此一 Web-based Botnet 偵測架構主要可分為流量收集模組、流量過濾模組、特徵選取模組

以及警訊模組，其整體架構如圖 7 所示。

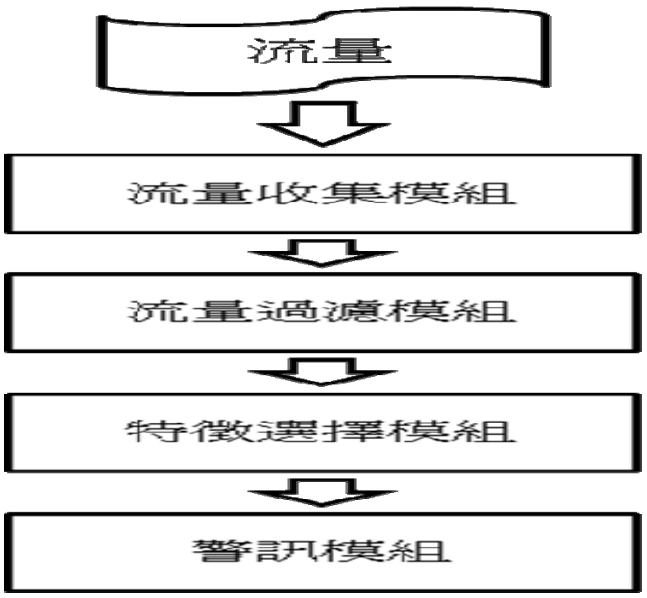


圖 7. Web-based Botnet 偵測架構

底下針對各個模組加以說明：

(1).流量收集模組：主要功用為收集 NetFlow 流量資訊。此一模組收集 Netflow V5 資訊，將其導入至資料收集主機後，利用 NFDUMP 將 NetFlow 儲存成 nfdump 格式，而後將其匯入資料庫，以供後續流量計算之用。紀錄至資料庫之欄位包含 Date_Start、Date_End、Duration、SrcIP、SrcPort、DstIP、DstPort、Protocol、Flags、Flows、Packets、Bytes、Bpp、Bps、Pps。一般而言，在 Router 或 Switch 上導出 NetFlow 流量資訊。

(2).流量過濾模組：針對所欲計算之欄位，進行過濾動作，諸如過濾出 Port 80 與時間欄位等資訊。

(3).特徵選取模組：萃取 Timeslot 與 NetFlow 欄位近似等特性，以檢查電腦主機是否具有 Web-based Bot 特有之行為。接著，根據一連串特徵比對結果，以決定某電腦主機是否為 Web-based Botnet 之一員。用以驗證之特徵包含 Timeslot、NetFlow 欄位近似、B2S、S2B 以及 Group Analysis 等特性。

(4).警訊模組：經由特徵選取模組比對 Web-based Botnet 之存在，發送警告訊息(Alert)通知網路管理人員監控受感染之電腦主機，並採取處理措施。

圖 8 為 Web-based Botnet 偵測流程：

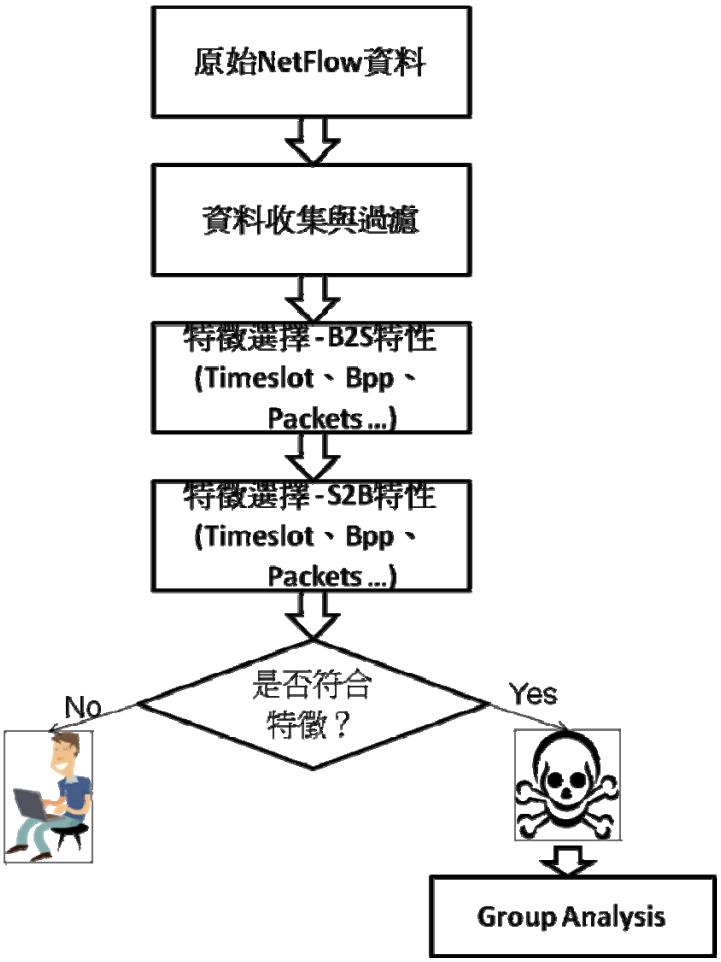


圖 8. Web-based Botnet 偵測流程

首先，收集 NetFlow 流量資訊並儲存至資料庫中，同時過濾出所需之欄位資料以供後續分析之用。接著，從資料庫中取出所需欄位資料，包含每筆 Flow 之間隔時間、Duration、Packets、Bytes、Bpp 等資料，計算其平均數與標準差。隨後，檢查電腦主機有無符合 Web-based Botnet 之特徵，諸如 Timeslot 之標準差是否小於設定值，以及 Packets、Bytes、與 Bpp 之值是否小於設定值，此階段主要檢查 B2S 特性。接著，將可疑之 Flow 儲存，繼續檢查 S2B 特性，從可疑之流量中計算 S2B 特性。若有 Flow 同時符合 B2S 與 S2B 特性，則確定為 Web-based Bot。最後，將

確定為 Web-based Bot 之 Flow 儲存，進行 Group Analysis，進一步檢查網路中有無存在多隻相同 Web-based Bot 形成之 Botnet。

由上述 Web-based Botnet 偵測流程可知，判斷 Web-based Bot 不能只靠單一特徵判斷，必須經由一連串特徵比對，諸如檢查電腦主機與某 Server 有無規律之連線特性，查核該 NetFlow 欄位資料之平均數與標準差，是否符合 B2S 與 S2B 特性，以及進行 Group Analysis，符合者方能確認其為 Web-based Bot。若單符合其中一項特發出則發出可疑警告。

五、Web-based Botnet 偵測實驗與分析

本研究經由多次實驗，並以不同 Bot 設定參數、不同網路拓樸、以及不同連線時間與收集時間，以驗證偵測方法之準確性。實驗設計方面，主要針對三種不同網路環境進行 Web-based Botnet 偵測實驗，一為「區網內存在一種 Botnet」，即環境中只存在一隻 Bot，並感染多台電腦; 二為「區網內存在多種 Botnet」，即環境中存在多種設定之 Bot，並連向不同 Web Server；最後，將 Testbed 之實驗情境轉移到真實區域網路進行，以確認在真實環境下是否可真正偵測出 Web-based Bot。

(一) 實驗情境 1-區網內存在一種 Botnet

實驗情境 1 之網路拓樸如圖 9 所示：

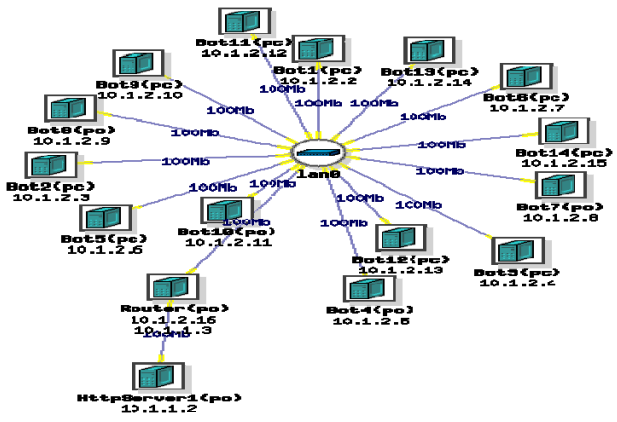


圖 9. 實驗情境 1-區網內存在一種 Botnet

在 Testbed 可用節點數有限下，為模擬多台 Bots，將角色簡化為只存在 Bots 與 Web Server，並將 Bot 設定為每分鐘與 Web Server 連線。實驗節點共包含 13 台 Bots 與 1 台 Web Server，並以 Router 串接。電腦主機感染為 Bots 後，透過 Router 與 Web Server 連線並取得指令。實驗過程中亦變換 Bots 連線時間與指令，以符合真實情況。

Bot 在感染後大部分時間為"潛伏期"，即與 Web Server 間之連線，除定時連線外，不進行其他活動，令使用者察覺其存在。在潛伏期中 Bot 所進行之動作呈現規律性，例如在一段時間內 (一個小時內) 與 Web Server 連線之 Timeslot 皆相同，取得之封包大小也相同。因此，只要掃描一段時間內每台電腦主機是否具有 B2S 與 S2B 特性，即可判斷是否為 Web-based Bot。

在本實驗情境中，掃描區段設為 1 小時。下列為 B2S 掃描結果之部份擷取:

(TimeStart, DstIP<=>SrcIP, Duration, Packets, Bytes, Bpp)
2009-06-26 14:02:49.784, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:03:49.791, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:04:49.806, 10.1.1.2<=>10.1.2.10, 60.036, 6, 533, 88
2009-06-26 14:05:49.842, 10.1.1.2<=>10.1.2.10, 60.017, 6, 533, 88
2009-06-26 14:06:49.859, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:07:49.874, 10.1.1.2<=>10.1.2.10, 60.005, 6, 533, 88
2009-06-26 14:08:49.880, 10.1.1.2<=>10.1.2.10, 60.007, 6, 533, 88
2009-06-26 14:09:49.887, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:10:49.902, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:11:49.908, 10.1.1.2<=>10.1.2.10, 60.027, 6, 533, 88
2009-06-26 14:12:49.935, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:13:49.950, 10.1.1.2<=>10.1.2.10, 60.026, 6, 533, 88
2009-06-26 14:14:49.976, 10.1.1.2<=>10.1.2.10, 60.027, 6, 533, 88
2009-06-26 14:15:50.004, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:16:50.018, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:17:50.024, 10.1.1.2<=>10.1.2.10, 60.007, 6, 533, 88
2009-06-26 14:18:50.031, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:19:50.046, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:20:50.052, 10.1.1.2<=>10.1.2.10, 60.007, 6, 533, 88
2009-06-26 14:21:50.060, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:22:50.074, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:23:50.080, 10.1.1.2<=>10.1.2.10, 60.007, 6, 533, 88
2009-06-26 14:24:50.087, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:25:50.102, 10.1.1.2<=>10.1.2.10, 60.016, 6, 533, 88
2009-06-26 14:26:50.118, 10.1.1.2<=>10.1.2.10, 60.017, 6, 533, 88
2009-06-26 14:27:50.135, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:28:50.151, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:29:50.167, 10.1.1.2<=>10.1.2.10, 60.016, 6, 533, 88
2009-06-26 14:30:50.184, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:31:50.199, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:32:50.215, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:33:50.222, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:34:50.237, 10.1.1.2<=>10.1.2.10, 60.005, 6, 533, 88
2009-06-26 14:35:50.243, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:36:50.250, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:37:50.265, 10.1.1.2<=>10.1.2.10, 60.005, 6, 533, 88
2009-06-26 14:38:50.271, 10.1.1.2<=>10.1.2.10, 60.016, 6, 533, 88
2009-06-26 14:39:50.287, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88

2009-06-26 14:40:50.303, 10.1.1.2<=>10.1.2.10, 60.005, 6, 533, 88
2009-06-26 14:41:50.309, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:42:50.316, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:43:50.331, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:44:50.347, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:45:50.354, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:46:50.369, 10.1.1.2<=>10.1.2.10, 60.005, 6, 533, 88
2009-06-26 14:47:50.375, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:48:50.382, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:49:50.396, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:50:50.402, 10.1.1.2<=>10.1.2.10, 60.007, 6, 533, 88
2009-06-26 14:51:50.410, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:52:50.424, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:53:50.430, 10.1.1.2<=>10.1.2.10, 60.007, 6, 533, 88
2009-06-26 14:54:50.438, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:55:50.453, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
2009-06-26 14:56:50.468, 10.1.1.2<=>10.1.2.10, 60.007, 6, 533, 88
2009-06-26 14:57:50.476, 10.1.1.2<=>10.1.2.10, 60.014, 6, 533, 88
2009-06-26 14:58:50.490, 10.1.1.2<=>10.1.2.10, 60.006, 6, 533, 88
2009-06-26 14:59:50.498, 10.1.1.2<=>10.1.2.10, 60.015, 6, 533, 88
From 2009-06-26 14:00:00.000
To 2009-06-26 15:00:00.000
SrcIP : 10.1.2.10
DstIP : 10.1.1.2
DstPt : 80
Amount of Flows : 58
Time_intervel(sec) => (AVG, SD) : (60.018, 0.133)
Packets => (AVG, SD) : (6, 0)
Bytes => (AVG, SD) : (533, 0)
Bpp => (AVG, SD) : (88, 0)

從上列掃描結果可知，掃描區間為 2009-06-26 下午 2 點至 3 點間 1 個小時之掃描紀錄，Source IP 為 10.1.2.10，Destination IP 為 10.1.1.2，共有 58 筆 Flow 經過計算，得到之 Timeslot 之平均數及標準差分別為 60.018 與 0.133，Bpp 之平均數及標準差分別為 88 與 0，正符合 B2S 特性。接著，反向檢查 S2B 的數值，如下列所示：

(TimeStart, DstIP<=>SrcIP, Duration, Packets, Bytes, Bpp)
2009-06-26 14:02:49.784, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:03:49.791, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:04:49.806, 10.1.2.10<=>10.1.1.2, 60.036, 4, 435, 108
2009-06-26 14:05:49.842, 10.1.2.10<=>10.1.1.2, 60.017, 4, 435, 108
2009-06-26 14:06:49.859, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:07:49.874, 10.1.2.10<=>10.1.1.2, 60.005, 4, 435, 108
2009-06-26 14:08:49.880, 10.1.2.10<=>10.1.1.2, 60.007, 4, 435, 108
2009-06-26 14:09:49.887, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:10:49.902, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:11:49.908, 10.1.2.10<=>10.1.1.2, 60.027, 4, 435, 108
2009-06-26 14:12:49.935, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:13:49.950, 10.1.2.10<=>10.1.1.2, 60.026, 4, 435, 108
2009-06-26 14:14:49.976, 10.1.2.10<=>10.1.1.2, 60.027, 4, 435, 108
2009-06-26 14:15:50.004, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:16:50.018, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:17:50.024, 10.1.2.10<=>10.1.1.2, 60.007, 4, 435, 108
2009-06-26 14:18:50.031, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:19:50.046, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:20:50.052, 10.1.2.10<=>10.1.1.2, 60.007, 4, 435, 108
2009-06-26 14:21:50.060, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:22:50.074, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:23:50.080, 10.1.2.10<=>10.1.1.2, 60.007, 4, 435, 108
2009-06-26 14:24:50.087, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:25:50.102, 10.1.2.10<=>10.1.1.2, 60.016, 4, 435, 108
2009-06-26 14:26:50.118, 10.1.2.10<=>10.1.1.2, 60.017, 4, 435, 108
2009-06-26 14:27:50.135, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:28:50.151, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:29:50.167, 10.1.2.10<=>10.1.1.2, 60.016, 4, 435, 108
2009-06-26 14:30:50.184, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:31:50.199, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:32:50.215, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:33:50.222, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:34:50.237, 10.1.2.10<=>10.1.1.2, 60.005, 4, 435, 108
2009-06-26 14:35:50.243, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:36:50.250, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:37:50.265, 10.1.2.10<=>10.1.1.2, 60.005, 4, 435, 108
2009-06-26 14:38:50.271, 10.1.2.10<=>10.1.1.2, 60.016, 4, 435, 108
2009-06-26 14:39:50.287, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:40:50.303, 10.1.2.10<=>10.1.1.2, 60.005, 4, 435, 108
2009-06-26 14:41:50.309, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:42:50.316, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:43:50.331, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:44:50.347, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:45:50.354, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:46:50.369, 10.1.2.10<=>10.1.1.2, 60.005, 4, 435, 108
2009-06-26 14:47:50.375, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:48:50.382, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:49:50.396, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:50:50.402, 10.1.2.10<=>10.1.1.2, 60.007, 4, 435, 108
2009-06-26 14:51:50.410, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:52:50.424, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:53:50.430, 10.1.2.10<=>10.1.1.2, 60.007, 4, 435, 108
2009-06-26 14:54:50.438, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:55:50.453, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
2009-06-26 14:56:50.468, 10.1.2.10<=>10.1.1.2, 60.007, 4, 435, 108
2009-06-26 14:57:50.476, 10.1.2.10<=>10.1.1.2, 60.014, 4, 435, 108
2009-06-26 14:58:50.490, 10.1.2.10<=>10.1.1.2, 60.006, 4, 435, 108
2009-06-26 14:59:50.498, 10.1.2.10<=>10.1.1.2, 60.015, 4, 435, 108
From 2009-06-26 14:00:00.000
To 2009-06-26 15:00:00.000
SrcIP : 10.1.1.2
DstIP : 10.1.2.10
DstPt : 80
Amount of Flows : 58
Time_intervel(sec) => (AVG, SD) : (60.018, 0.133)
Packets => (AVG, SD) : (4, 0)
Bytes => (AVG, SD) : (435, 0)
Bpp => (AVG, SD) : (108, 0)

從上列掃描結果可知，掃描區間亦為 2009-06-2 下午 2 點至 3 點 1 個小時之掃描紀錄，Source IP 為 10.1.1.2，Destination IP 為 10.1.2.10，共有 58 筆 Flow 經過計算，得到之 Timeslot 之平均數及標準差分別為 60.018 與 0.133，Bpp 之平均數及標準差分別為 108 與 0，亦符合 S2B 特性。

經由 B2S 與 S2B 特性檢查後，再將所有 Bot 資料進行 Group Analysis，以判斷是否為同一群 Botnet。下列為 Group Analysis 掃描紀錄：

DstIP : 10.1.1.2
Total SrcIP : 27
From:2009-06-26 14:00:00.000
To :2009-06-26 16:00:00.000
==========
(AVG, SD)
ScanDate, SrcIP, Timeslot, Packets, Bpp, Flow
2009-06-26 14:00:00.000, 10.1.2.10, (60.018, 0.133), (6, 0), (88, 0), 58

2009-06-26 15:00:00.000, 10.1.2.10, (60.021, 0.145), (5.98, 0.142), (88.184, 1.273), 49
2009-06-26 14:00:00.000, 10.1.2.11, (60.018, 0.133), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.11, (60, 0), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.12, (60.018, 0.133), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.12, (60, 0), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.13, (60.018, 0.133), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.13, (60, 0), (5.98, 0.142), (88.184, 1.273), 49
2009-06-26 14:00:00.000, 10.1.2.14, (60.018, 0.133), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.14, (60.021, 0.143), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.15, (60, 0), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.15, (60.084, 0.454), (5.98, 0.142), (88.184, 1.273), 49
2009-06-26 15:00:00.000, 10.1.2.2, (60, 0), (5, 0), (96, 0), 50
2009-06-26 14:00:00.000, 10.1.2.3, (60, 0), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.3, (60.021, 0.143), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.4, (60.053, 0.398), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.4, (60.021, 0.143), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.5, (60.018, 0.133), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.5, (60, 0), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.6, (60.018, 0.133), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.6, (60, 0), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.7, (60, 0), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.7, (60.021, 0.143), (5.98, 0.14), (88.18, 1.26), 50
2009-06-26 14:00:00.000, 10.1.2.8, (60, 0), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.8, (60.021, 0.143), (5.96, 0.28), (88.44, 3.08), 50
2009-06-26 14:00:00.000, 10.1.2.9, (60, 0), (6, 0), (88, 0), 58
2009-06-26 15:00:00.000, 10.1.2.9, (60.021, 0.143), (5.98, 0.14), (88.18, 1.26), 50

由上列掃描結果可知，共有 13 個 IP (10.1.2.2-15) 連向相同之 Destnation IP (10.1.1.2)，其 Timeslot 特性與 NetFlow 欄位計算值亦非常相近，由此可判斷為同一種 Botnet。

(二) 實驗情境 2-區網內存在多種 Botnet

實驗情境 2 設定 3 種 Web-based Botnet，以及 3 台 Web Server，其設定連線時間分別為 1 分鐘、10 分鐘、與 15 分鐘。實驗情境 2 之網路拓樸如圖 10 所示。

B2S 之掃描區間為 2009-06-16 凌晨 3 點至 4 點間 1 個小時之掃描紀錄，Source IP 為 10.1.4.5，Destination IP 為 10.1.1.4，共有 12 筆 Flow 經過計算，得到 Timeslot 之平均數及標準差分別為 300.091 與 0.301，Bpp 之平均數及標準差分別為 87.5 與 01.659，正符合 B2S 特性。接著，反向檢查 S2B 數值。其掃描區間亦為 2009-06-16 凌晨 3 點至 4 點間 1 個小時之掃描紀錄，Source IP 為 10.1.1.4，Destination IP 為 10.1.4.5，共有 12 筆 Flow 經過計算，得到 Timeslot 之平均數及標準差分別為 300.091 與 0.301，Bpp 之平均數及標準差分別為 106 與 3.594，正符合 S2B 特性。經由 B2S 與 S2B 特性檢查後，再將所有 Bot 資料進行 Group Analysis，以判斷是否為同一群 Botnet。Group Analysis 結果顯示共有 10 台 Bots 連向 3 台不同 Web Server，其中 10.1.4.10、10.1.4.11 連向 10.1.1.2，10.1.4.6-9 連向 10.1.1.3，而 10.1.4.2-5 連向 10.1.1.4，由此可區別出三種不同設定之 Web-based Botnet。

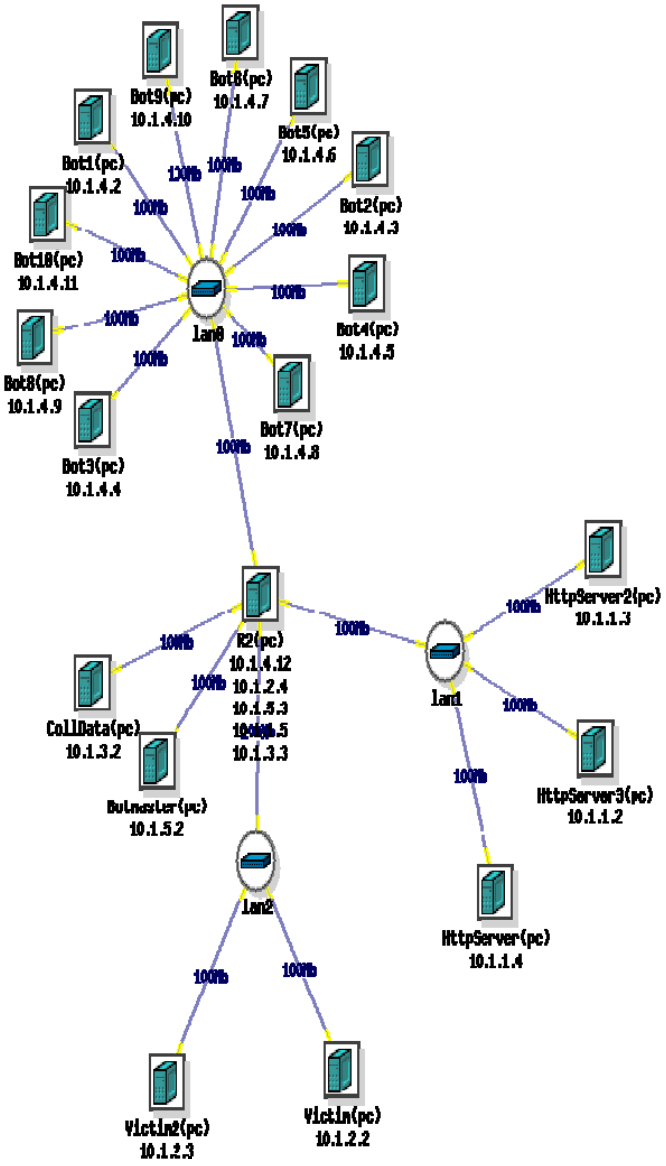


圖 10. 實驗情境 2-區網內存在多種 Botnet

(三) 真實區域網路資料驗證

在此環境中共有約 20 台真實主機，其中一台主機將其感染為 Bot，其 IP 為 140.117.241.195; 而另一台主機扮演 Web Server 角色，並提供 Bot 連線取得指令，其 IP 為 140.117.241.205。實驗環境設定完成後，在 Switch 導出區域網路流量，並進行 Web-based Botnet 偵測。

底下描述於真實區域網路環境下之實驗結果：

(1).掃描區間為 2009-07-15 8 點至 9 點間 1 個小時之掃描紀錄，Source IP 為 140.117.241.195，Destination IP 為 140.117.241.205，共有 33 筆 Flow 經過計算，得到 Timeslot 之平均數及標準差分別為 60.032 與 0.177，Bpp 之平均數及標準差分別為 111.667 與 3.772，正符合 B2S 特性。

(2.)S2B 掃描結果，發現 140.117.241.195 與 140.117.241.205 此兩台主機符合 B2S 與 S2B 特性，確認為其為 Web-based Bot，並與實驗所設定之 Bot 相符

(3).以目前實驗結果來看，在真實環境下利用 B2S 與 S2B 特性，進行 Web-based Botnet 偵測為可行之方法。

由於此實驗僅在 20 台主機之區域網路下進行，未來若可在大型網路環境下，持續一段時間進行實驗，可驗證本研究提出之偵測 Web-based Botnet 方法之有效性與準確性。

六、結論

殭屍網路的危害日趨嚴重，從 IRC-based Botnet、P2P Botnet 到 Web-based Botnet，每一種進化與演變，都以更迅速的擴張與更難以被偵測之宗旨不斷演進。其中 Web-based Botnet 將本身隱藏在 HTTP 中，與正常流量混合在一起，難以偵測其存在。有鑑於此，本研究希望找出 Web-based Botnet 可能共有之特性，以供偵測 Web-based Botnet 或日後更進一步之研究。

本研究藉由建立真實實驗環境，並透過多次實驗，不同 Bot 參數設定，並模擬真實之感染及攻擊流程，進一步收集流量與 Payload，以進行封包分析與研究，並透過 NetFlow 彙整之資訊，進行驗證。本研究發現 Blackenergy 有定時連線 (Timeslot 特性)、NetFlow 欄位值近似、B2S (Bot to Server) 與 S2B (Server to Bot)等特性，成功利用上述特性偵測 Web-based Botnet，而且將相同特性之 Bots 歸類為同一群 Web-based Botnet。同時，搭配不同實驗環境設定以進行實驗，更可提高偵測方法之可信度與準確性。

本研究所著重之 Bot 為 Blackenergy，偵測所使用之特徵對於整個 Web-based Botnet 可能稍嫌不足。從 Team Cymru 之研究發現，如 machbot 此種 Web-based Botnet 亦有定時連向同一頁面，並取得指令之特性，正與 Blackenergy 之習性相似。因此，期望日後能利用本研究所發現之特性，對其他種類 Web-based Bots 進行實驗與偵測，期望取得更多特徵屬性，以偵測 Web-based Botnet，進而使偵測技術更為完善。同時，未來若能實際於大型網路下(諸如學校宿舍網路、企業網路) 持續一段時間進行實驗，以瞭解在複雜之大型網路下使用本研究所發現之特性是否能正確偵測 Web-based Botnet，以及偵測效能為何，或可進一步發現其他特性以杜絕 Web-based Botnet 之擴散。

致謝

This work was supported in part by Testbed@TWISC, National Science Council under the Grants NSC 98-2219-E-006-001.

參考文獻

[1] P. Barford and V. Yegneswaran, “An Inside Look at Botnets,” Special Workshop on

Malware Detection, Advances in Information Security, Springer Verlag, 2006.

[2] Cisco IOS NetFlow, Available from: http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[3] E. Cooke, F. Jahanian, and D. McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets," Proceedings of the Steps to Reducing Unwanted Traffic on the Internet, pp. 39-44, 2005.

[4] F.C. Freiling, T. Holz, and G. Wicherski, "Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks," Lecture Notes in Computer Science, vol. 3679, pp. 319-335, 2005.

[5] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation," Proceedings of the 16th USENIX Security Symposium, August, 2007.

[6] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," Proceedings of the 15th Annual Network and Distributed System Security Symposium, February, 2008.

[7] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale Botnet Detection and Characterization," HotBots'07 First Workshop on Hot Topics in Understanding Botnets, April 10, 2007.

[8] J.S. Lee , H.C. Jeong , J.H. Park , M. Kim , and B.N. Noh, "The Activity Analysis of Malicious HTTP-Based Botnets Using Degree of Periodic Repeatability", International Conference on Security Technology 2008, pp. 83-86, 2008.

[9] C. Livadas, B. Walsh, D. Lapsley, and W.T. Strayer, "Using Machine Leaning Techniques to Identify Botnet Traffic," Proceedings of 31st IEEE Conference on Local Computer Networks, pp. 967-974, 2006.

[10] NFDUMP, Available from: http://nfdump.sourceforge.net/.

[11] NFSen, Available from: http://nfsen.sourceforge.net/.

[12] M.A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A Multifaceted Approach to Understanding the Botnet Phenomenon," Internet Measurement Conference 2006, October 25-27, 2006.

[13] Snort, Available from: http://www.snort.org.

[14] W.T. Strayer,, B. Walsh, C. Livadas, and D. Lapsley, "Detecting Botnets with Tight Command and Control," Proceedings of 31st IEEE Conference on Local Computer Networks, pp. 195-202, 2006.

[15] Tcpdump, Available from: http://www.tcpdump.org/.

[16] Team Cymru, "A Taste of HTTP Botnets," July 2008, Available from: http://www.team-cymru.org/ReadingRoom/Whitepapers/2008/http-botnets.pdf.

[17] Testbed @ NCKU, Available from: https://testbed.ncku.edu.tw.

[18] Wireshark, Available from: http://www.wireshark.org.