淺談軟體定義網路於分散式阻斷服務攻擊之應用

作者／陳國鐘、歐淙富、顏佳冠

提要

一、近幾年來隨著網路的發達，網路安全問題也隨之增加，點對點殭屍網路(Peer to Peer Botnet, P2P Botnet) 造成的資安問題更是不計其數，其中最常見的攻擊便是透過殭屍電腦所進行的分散式阻斷服務攻擊(Distributed Denial of Service, DDoS)。

二、隨著雲端技術逐漸成熟與市場的大量導入，傳統網路架構與框架不論是在佈建、設定與維護上均無法趕上彈性、靈活與快速的新IT環境需求。因此，軟體定義網路(Software-Defined Networking, SDN)便應運而生。

三、本研究主要探討如何透過軟體定義網路結合機器學習(Machine Learning)資料分析技術，以偵測分析並進一步阻止分散式阻斷服務攻擊之行為，確保雲端服務之正常運作。

關鍵詞：殭屍電腦、分散式阻斷服務攻擊、軟體定義網路、機器學習。

前言

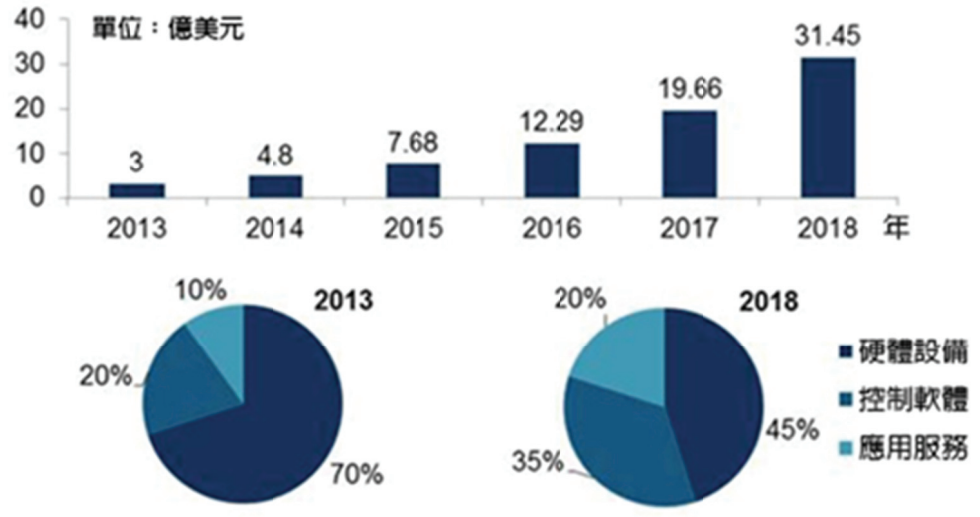
在網際網路蓬勃發展的時代，網際網路上的資安攻擊事件越來越多，其中大部分的駭客透過大量的殭屍電腦針對特定的目標進行分散式阻斷服務攻擊，最近較有名的例子為香港大學的民意網站(PopVote)投票系統、壹傳媒的網站遭受到分散式阻斷服務攻擊，主要的攻擊手法就是利用大量的殭屍電腦，短時間內攻擊特定的目標，使得伺服器無法提供正常的服務。

另一方面，根據工研院 IEK 預估，2013 年軟體定義網路硬體設備、控制軟體加應用服務的產值將超過 3 億美元，並可望於 2018 年大幅成長至 31 億 4,500 萬美元，2013～2018 年的產值年複合成長率(CAGR)將高達 60%(如圖一)，儼然已躍居網通產業新一代巨星。

目前最主流的軟體定義網路的實現方法首推 OpenFlow，OpenFlow 一改傳統網路的舊習，將原本由硬體路由器(Router)決定的資料傳送路徑改為由 OpenFlow 控制器(Controller)產生，由於傳統網路各硬體自主決定資料傳送路徑，若原有路徑失敗則造成大量的重複資料傳輸，造成對系統資源的浪費及系統效能的損害，若需修改時也要網路管理人員逐一手動更新硬體內之軟體，耗費大

量人力資源及企業成本，而且透過人工逐一設定的方式也有很高的風險，一旦網路管理人員輸入了錯誤的指令，很容易造成網路服務癱瘓，OpenFlow 將傳輸路徑交由控制器全盤規劃，可視需求動態改變達成最佳化，需修改時也僅需要修改控制器的規則，大量節省了不必要的資源浪費。

圖一 2013～2018年SDN市場產值預測



資料來源：黃耀瑋，〈電信/網路營運商力拱 SND布建商機全面引爆〉《新通訊》，http://www.2cm.com.tw/coverstory_content.asp?sn=1310040003，西元2016年1月4日。

因此，本研究將利用軟體定義網路架構，結合資料分析資安防護機制，以機器學習方式分析網路流量，辨識哪些主機可能是透過點對點(Peer to Peer , P2P)模式發出攻擊的殭屍電腦，並進一步藉由 SDN 控制器(Controller)對 OpenFlow 交換器(Switch)下達適當的指令，快速阻隔網路攻擊流量，保護整個雲端服務的安全。

分散式阻斷服務攻擊

隨著網路發達，許多 P2P 軟體可以透過各種相關協定在網路之間有效率地進行傳輸。然而，透過 P2P 架構來部署的殭屍網路所產生的問題也在近幾年快速增加。為能有效率又安全地管理所負責的網路環境底下的網路流量，網路管理人員必須耗費相當大的心力與成本來辨識、偵測、管理與分類 P2P 傳輸的網路流量，這對網路管理人員是相當重要，卻也相當龐大的任務。

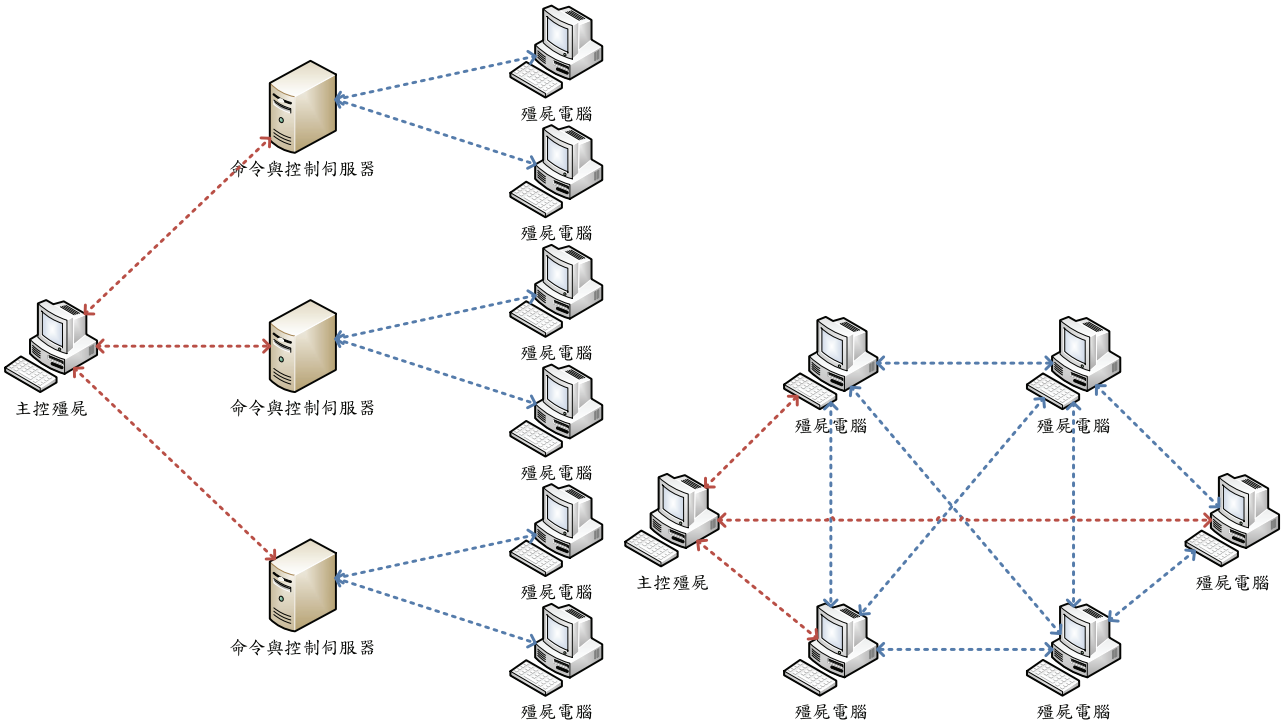
殭屍網路(Botnet) 通常是一群大量散落在各地受感染的機器所組成，因此擁有分散式的架構及龐大的運算資源，通常會被一個主控殭屍(Botmaster)透過命令

與控制伺服器(Command and Control Server, C&C Server)來管理並接受指令進行各種惡意攻擊或非法的行為，例如發動分散式阻斷服務攻擊、散布垃圾電子郵件、竊取敏感的個人資料，或利用殭屍網路的運算能力進行分散式運算以達到某種特殊目的。因此，殭屍網路在近幾年儼然已將成為網路安全中最嚴重的問題，這也顯現了防禦與偵測殭屍網路行為在現今網路安全領域中的重要性。

一、殭屍網路

殭屍網路發展數十載，因為網際網路中繼聊天(Internet Relay Chat, IRC)與超文本傳輸協定(HyperText Transfer Protocol, HTTP)的使用簡單方便而普及。所以，IRC和HTTP/HTTPS曾是命令與控制傳達階段最主要及受歡迎的協定，並被主控殭屍廣泛運用來安排及部署他們的殭屍網路，然而這類型的殭屍網路也可以稱為集中式殭屍網路(Centralized Botnet)，是基於高度集中式的網路架構，但是這種架構有很大的機會受到單一節點失敗(Single Point of Failure) 而被擊潰，並且在這種架構中C&C Server也會因為大量的訊息交換而容易被查覺及防禦，網路安全人員一旦找到C&C Server就可以輕易的瓦解IRC或HTTP/HTTPS的殭屍網路，圖二(a)顯示了集中式殭屍網路的網路拓樸(Network Topology)，它們擁有一個集中式的C&C Server網路架構，所有殭屍網路中的殭屍電腦將透過一或多個C&C Server受主控殭屍控制。

圖二 殭屍網路架構



(a) 集中式網路　　(b) 分散式網路

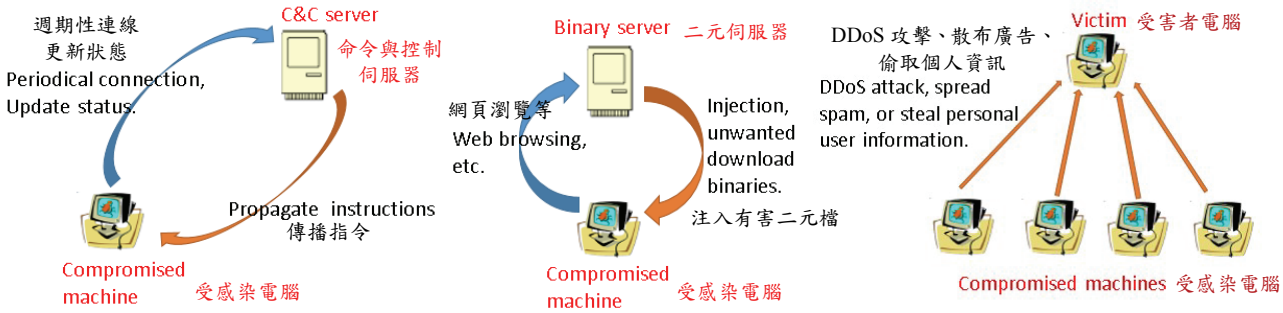
資料來源：本研究繪製

而近幾年，由於單一節點失敗的缺點，殭屍網路的溝通已逐漸轉型為P2P架構，主控殭屍可以利用P2P的通訊協定來溝通與部署殭屍網路，這種P2P殭屍網路的架構中沒有集中式的C&C Server，取而代之的是每個受感染的機器，都是主端也是客端，除了接受指令以外，也可以當做中間的媒介傳送攻擊指令，沒有了集中式的控制架構，主控殭屍可以利用任一台殭屍電腦來傳送指令或散播惡意檔案給其他殭屍電腦，或是從這些殭屍電腦中獲取有用的資訊。分散式的點對點殭屍網路架構如圖二(b)所表示，在這種架構之下，主控殭屍可以在任何位置及任何時間傳送指令發動惡意行為，相較於集中式的殭屍網路，點對點殭屍網路有較高的彈性與韌性，就算當其中一台殭屍電腦被擊潰，其它的殭屍電腦與主控殭屍依然可以維持原本的連線與溝通，繼續進行惡意行為，這也顯示了防禦與偵測點對點殭屍網路，更需要被重視及進一步研究。

二、殭屍網路生命週期

殭屍網路的生命週期已被許多學者提出來，像是Leonard et al.[1]以及Silva et al.[2]對殭屍網路生命週期的定義大致相同，分為三個階段：感染階段(Infection Stage)、命令與控制傳達階段(C&C Communication Stage)和攻擊階段(Attack Stage) ，如圖三。

圖三 殭屍網路生命週期



(a) 感染階段　(b) 命令與控制傳達階段　(c) 攻擊階段

資料來源：Sherif Saad, Issa Traore et al. 2011 PST,
(Ninth Annual International Conference on Privacy, Security and Trust)

在感染階段，主控殭屍會透過釣魚網站或者社交工程來感染網路上的其它電腦，這些受害機器通常在不知情的情況下開啟了受感染的電子郵件或檔案，並從一個二元伺服器(Binary Server)中下載了殭屍代碼(Bot Code)執行，因而成為殭屍網路中的一員；在第二階段，通常主控殭屍會透過C&C Server對遭受感染的

[1] J. Leonard, S. Xu, and R. Sandhu, "A Framework for Understanding Botnets," In Proceedings of the International Workshop on Advances in Information Security, Fukuoka Institute of Technology, 2009.
[2] S. S. Silva, R. M. Silva, R. C. Pinto, R. M. Salles, "Botnets: A survey," In Computer Networks, vol. 57, no.2, 2013, pp. 378-403.

殭屍電腦更新惡意軟體代碼(Malware Code)或傳送指令，而這些殭屍電腦也會定期地向C&C Server連線回報狀態及確認彼此存在，殭屍電腦與C&C Server中間的溝通可能會透過不同的協定來做連線行為；最後一個階段是攻擊階段，在這個階段中，受到主控殭屍控制的殭屍電腦會依照接受到的指令進行各式各樣的惡意行為，例如發動DDoS攻擊、散布垃圾郵件，竊取受害電腦中的資訊，攻擊或找尋其他受害的機器。

三、殭屍網路偵測方式

為了能夠減低或避免由殭屍網路所產生的各種資訊安全威脅，過去數十年已有許多研究基於不同技術與架構提出他們的偵測方法[3][4]，這些研究基於各種不同的主要技術，以及對於殭屍電腦之間的行為假設來設計偵測與分析殭屍網路的方式。然而在這些眾多傑出的研究當中，其中基於機器學習方法來偵測殭屍網路的研究顯得相當優秀與突出[5]，Hu et al.[6]研究快速變遷為殭屍網路(Fast-Flux Botnet)的特性，並找出特徵利用多層支持向量機(Support Vector Machine, SVM)演算法來偵測快速變遷殭屍網路；Bilge et al.[7]利用分析 NetFlow[8]中取出的多種功能並建造基於不同演算法的模型來偵測殭屍網路中的C&C Server；在Saad et al.[9]與 Stevanovic et al.[10]的研究中，作者實驗多種不同的機器學習演算法對於偵測殭屍網路的表現，並在實驗結果得到很高的偵測率。

這些基於機器學習的方法基本假設是殭屍網路在溝通之中會產生和正常網路流量相當不同且具有可分辨性的網路連線行為，藉由這些差異，我們可以透過機器學習進行有效率地偵測與分析，而這類研究中的最大優點在於機器學習演算法能幫助我們找到殭屍網路之間的溝通模式，並利用學習到的模式進行網路行為的分析與偵測，我們不必對所有網路封包或是機器中的資源進行深入的分析與觀測，也不需在事先對這些溝通模式有相當深入的觀察與分析才能偵測殭屍網路行為。

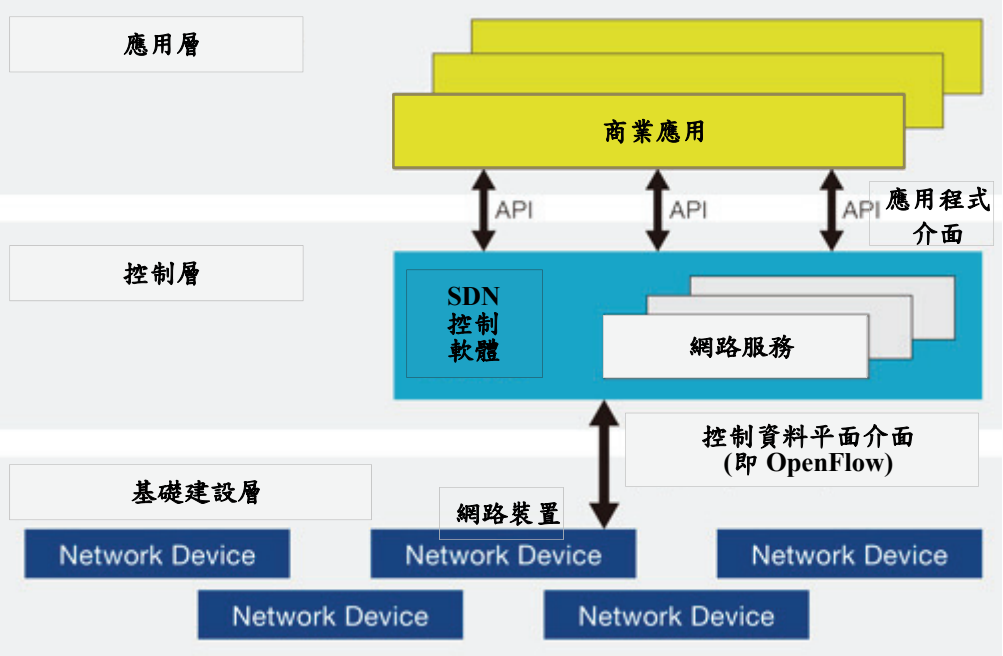
---

[3] M. Stevanovic, and J. M. Pedersen, ”An efficient flow-based botnet detection using supervised machine learning,” In Computing, Networking and Communications (ICNC), 2012 International Conference on, Feb 2014, pp.797-801.
[4] S. C. Guntuku, P. Narang, and C. Hota, ”Real-time Peer-to-Peer Botnet Detection Framework based on Bayesian Regularized Neural Network,” In CoRR, 2013, arXiv:1307.7464v1.
[5] M. Stevanovic and J. M. Pedersen, “Machine learning for identifying botnet network traffic,” Aalborg University, Technical Report, 2013.
[6] X. Hu, M. Knysz, and K.G. Shin, “Measurement and Analysis of Global IP-Usage Patterns of Fast-Flux Botnets,” In Proceedings of the 30th Annual IEEE International Conference on Computer Communications (INFOCOM), 2011.
[7] L. Bilge, D. Balzarotti, W. K. Robertson, E. Kirda, and C. Kruegel, ”Disclosure: detecting botnet command and control servers through large-scale netflow analysis,” In ACSAC, 2012.
[8] P. Phaal, ”sFlow Specification Version 5,” July 2004.
[9] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, ”Detecting P2P botnets through network behavior analysis and machine learning,” In Proceedings of 9th annual conference on privacy, security and trust (PST’11), 2011.
[10] M. Stevanovic, and J. M. Pedersen, ”An efficient flow-based botnet detection using supervised machine learning,” In Computing, Networking and Communications (ICNC), 2012 International Conference on, Feb 2014, pp.797-801.

軟體定義網路

以技術的角度而言，SDN 打破傳統 TCP/IP 的概念，把一台網路交換器的控制平面(Control Plane)與資料平面(Data Plane)分開，讓交換器只單純負責資料平面的資料傳送工作，而另外獨立出一台或是數台的 SDN 控制器負責控制平面，即管理與規則派送工作，交換器只負責處理封包在埠口(Port)的交換，至於封包該從哪一個埠進入，哪一個埠送出則完全交給控制器來控制及處理。控制器中間南向介面(Southbound)，透過如 OpenFlow 協定與網路設備進行溝通；北向介面(Northbound)，提供應用程式介面(Application Programming Interface, API)實作客製化功能需求之服務，SDN 之網路架構如圖四。

圖四 SDN網路架構示意圖



資料來源：Open Networking Foundation While Paper, “Software-Defined Networking: The New Norm for Networks,” 2012/4/13.

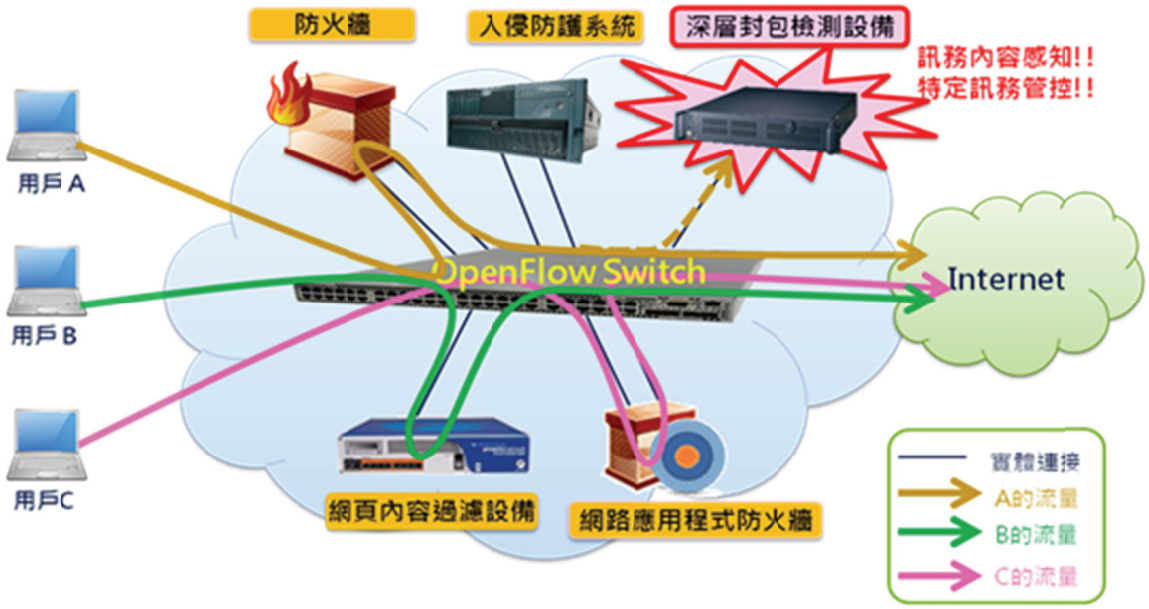
以應用服務的角度而言，企業可依據不同應用軟體的重要優先順序需求、實際網路運行的狀態等，動態指示控制器下達封包不同路徑走法，以滿足上述需求。完成 SDN 的架構後，所有的行為邏輯就是以「使用者、應用程式、服務」為主體，其他的基礎網路架構與運算元件都是配合上述三項需求而動態產生。

SDN 打破過去傳統網路的限制，提供可程式化的資料流控制，能夠動態地指定路由，進而提升網路頻寬的使用率。並藉由 SDN 網路的特性來提供雲端間的資源共享，將能有效改善雲端服務的達成率以及使用者之感受程度，直接或間接地讓雲端提供者獲益。

一、緣起與演進

2006 年 SDN 誕生於美國 GENI 專案資助的史丹佛大學(Stanford University)Nick McKeown 教授為首的研究團隊，該團隊提出了 OpenFlow 概念用於校園網路的試驗創新，後續基於 OpenFlow 給網路帶來可程式設計特性，SDN 的概念因應而生，其概念如圖五。

圖五 軟體定義網路概念



資料來源：中華電信研究院，〈數分 Hinet 資安加值服務架構優化〉，http://www.chttl.com.tw/web/ch/service/service_1_post8.html, 2015/12/16.

OpenFlow 規範已經歷了 1.1、1.2 以及 1.3 等版本， 1.4 標準也已在開放網路基金會 2013 年 6 月獲得批准發布。網路基金會成員 96 家，其中創建該組織的核心會員有 7 家，分別是 Google、Facebook、NTT、Verizon、德國電信、微軟、雅虎，其重要演進版本如圖六。

圖六 OPEN FLOW協議標準演進過程

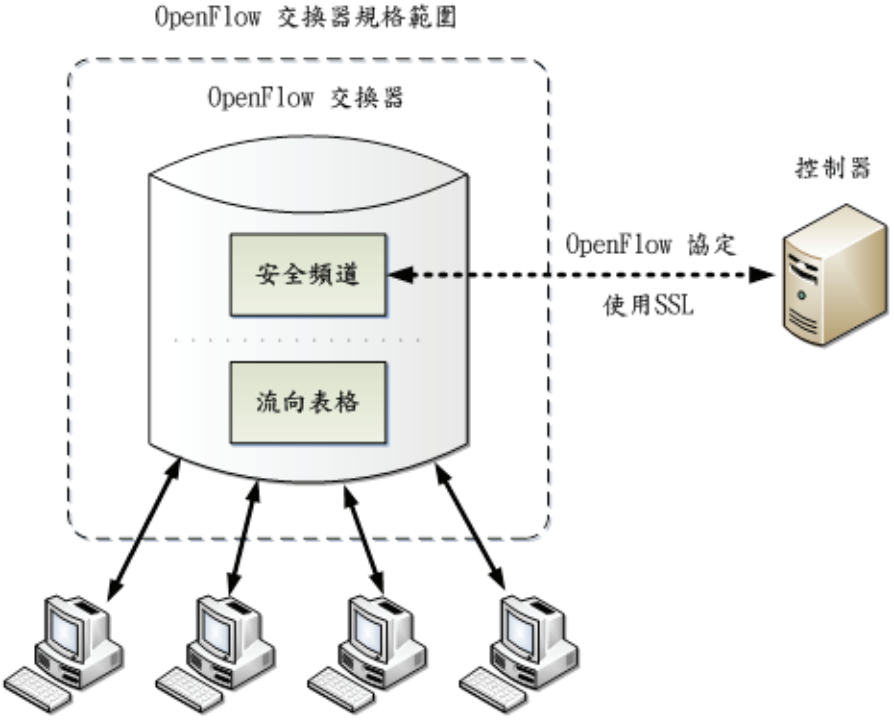
資料來源：C114中國通信網，〈OpenFlow協議標準演進過程〉，http://pad.c114.net/170/a842298.html，西元2015年12月16日。

二、OpenFlow

SDN 控制器就像是人類的大腦，統一下達指令給網路設備，網路設備則專

責於封包的傳遞，就像是人類的四肢負責執行各項動作，而 OpenFlow 技術則是一項通訊協定，用於控制層和資料層間建立傳輸通道，就像是人類的神經一樣，負責大腦與四肢的溝通，此協定目前也是實現 SDN 架構最主流的技術。

圖七 OpenFlow的原理和基本架構



資料來源：參考自The OpenFlow Switch Consortium, http://www.hightecnewstoday.com/jul-2011-high-tech-news-archives, 2015/12/16。

依據 Nick 等人的論文「OpenFlow：Enabling Innovation in Campus Networks [11]」，圖七常被用來說明 OpenFlow 的原理和基本架構。控制器與 OpenFlow 交換器(Siwtch)之間建立安全通道(Secure Channel)連接，通道間以 SSL 加密技術傳輸，以確保傳輸之間安全，使用者可使用 OpenFlow 協定提供開放原始碼與交換器硬體溝通，設定路由表(Flow Table)，例如經過哪些交換器，需要多少的網路頻寬，再藉由路由表定義封包傳送路徑。因為傳輸路徑已預先設定完成，交換器不需要透過不斷學習來尋找封包傳送的路徑，可大幅提升傳輸效率，降低延遲（Latency）的時間。

SDN對於分散式阻斷服務攻擊之應用

藉由 SDN 與 OpenFlow 協定這樣的新型態網路運作模式，可以幫助網路管理人員用軟體來控制與管理網路設備，並可以利用控制平面與資料平面分離的

[11] OpenFlow: Enabling Innovation in Campus Networls, http://www.openflow.org/documents/openflow-wp-latest.pdf, 2015/12/16.

特點達到許多在傳統網路環境中無法達到的網路服務與功能性。然而在 SDN 環境中控制平面透過 OpenFlow 協定只能處理到網路第四層(OSI Layer 4)的資料封包，沒有辦法直接對應用層(OSI Layer 7)做處理，在 OpenFlow 1.3 版本中，所有將近 40 個可以控制流向規則(Flow Rules)的匹配場值(Match Field Values)標示如表一。而且在 SDN 的環境中像殭屍網路等的安全問題依然存在[12]，甚至存在著更嚴重的資訊安全問題待解決。

本研究目的在 SDN 環境下利用機器學習分析網路封包，並透過分析後的結果藉由 OpenFlow 協定對交換器做控制，藉由修改流向規則來控管網路封包流向，做一個自動化的網路控管，減輕網管人員的工作，由於有 SDN 這樣的架構讓網路管理人員可以透過軟體撰寫控制層的功能，並藉此控制底層網路的封包處理與傳遞、繞送規則[13]。

表一 OPENFLOW 1.3流向配對表(OpenFlow 1.3 Flow Match Fields)

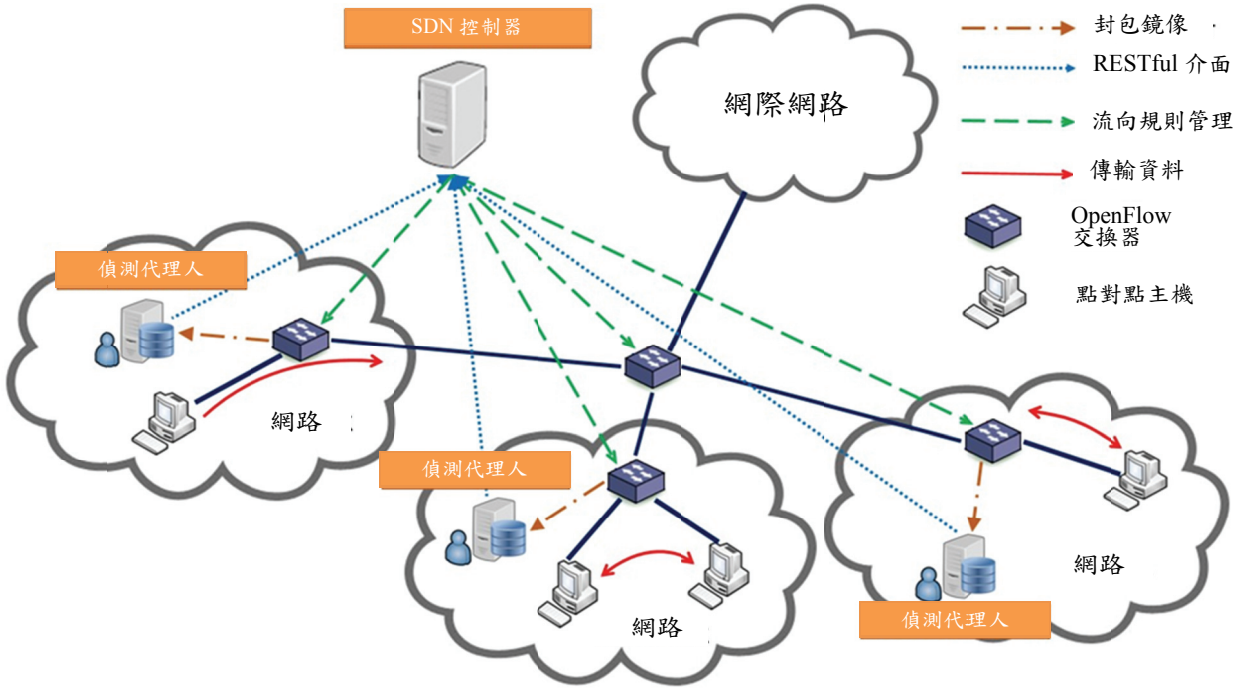
交換器輸入埠(Switch Input Port)	網際網路控制訊息通訊協定代碼(ICMP Code)
交換器實體輸入埠(Switch Physical Input Port)	位址解析通訊協定操作代碼(ARP (Address Resolution Protocol) Opcode)
表格間傳遞的元數據(Metadata Passed Between Tables)	位址解析通訊協定來源 IPv4 位址(ARP Source IPv4 Address)
乙太網路目的位址(Ethernet Destination Address)	位址解析通訊協定目標 IPv4 位址(ARP Target IPv4 Address)
乙太網路來源位址(Ethernet Source Address)	位址解析通訊協定來源硬體位址(ARP Source Hardware Address)
乙太網路框架類型(Ethernet Frame Type)	位址解析通訊協定目標硬體位址(ARP Target Hardware Address)
虛擬區域網路 ID (VLAN Id)	IPv6 來源位址(IPv6 Source Address)
虛擬區域網路優先權(VLAN Priority)	IPv6 目的位址(IPv6 Destination Address)
IP 差分服務代碼點(IP DSCP (Differentiated Services Code Point))	IPv6 流程標號(IPv6 Flow Label)
IP 明確擁塞通知(IP ECN(Explicit Congestion Notification))	ICMPv6 類型(ICMPv6 Type)
IP 協定(IP Protocol)	ICMPv6 代碼(ICMPv6 Code)
IPv4 來源位址(IPv4 Source Address)	鄰居發現協議使用目標位址(Target Address for ND(Neighbor Discovery))

[12] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-Defined Networks," In Proceedings of the second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking(HotSDN), 2013.
[13] N. McKeown, et al. "OpenFlow: Enabling Innovation in Campus Networks," In SIGCOMM Computer Communications Review, vol. 38, no.2, 2008.

IPv4 目的位址(IPv4 Destination Address)	鄰居發現協議使用來源鏈結層(Source link-layer for ND)
TCP 來源埠(TCP Source Port)	鄰居發現協議使用目標鏈結層(Target link-layer for ND)
TCP 目的位址(TCP Destination Address)	多重通訊協定標籤交換傳輸標籤(MPLS Label)
UDP 來源埠(UDP Source Port)	多重通訊協定標籤交換傳輸類別(MPLS TC(Traffic Class))
UDP 目的位址(UDP Destination Port)	多重通訊協定標籤交換堆疊底部位元(MPLS BoS(Bottom of the Stack) bit)
串流控制傳輸協議來源埠(SCTP(Stream Control Transmission Protocol) Source Port)	基於前綴的位元圖交集安全身份標識號碼 (Prefix-Based Bitmap Intersection Security ID, PBBI-SID)
串流控制傳輸協議目的埠(SCTP Destination Port)	邏輯埠元數據(Logical Port Metadata)
網際網路控制訊息通訊協定(ICMP(Internet Control Message Protocol) Type)	IPv6 延伸標頭虛擬場域(IPV6 Extension Header Pseudo-Field)

資料來源：本研究繪製

圖八 系統概觀



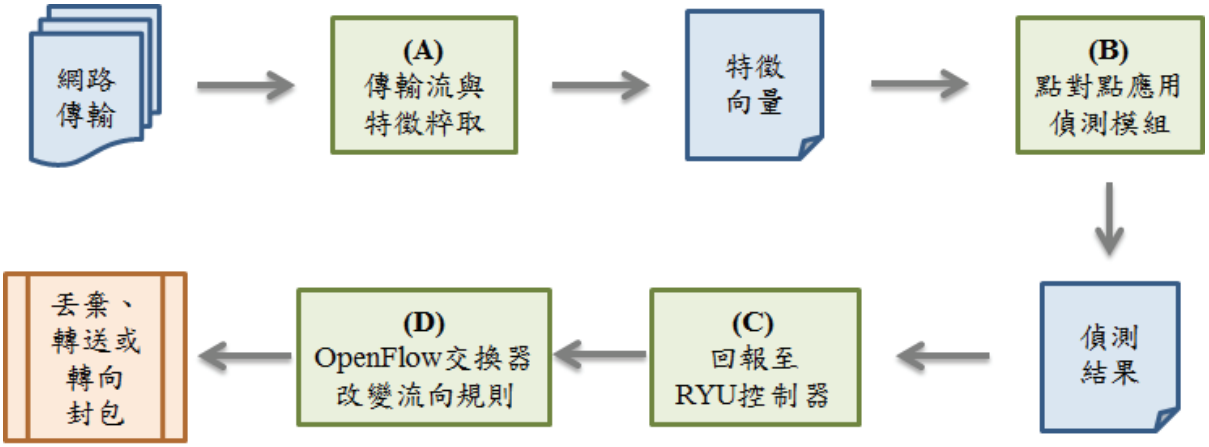
資料來源：本研究繪製

一、系統概觀

如圖八，首先我們在所有邊境的交換器上利用一個偵測代理人(Detection Agent)來收集並分析交換器下各端通過交換器的網路封包，偵測代理人收集到流

經的網路封包並匯集成傳輸流(Traffic Flow)後，再從收集到的 Flow 中取出多種特徵，利用訓練好的機器學習模型(Machine Learning Model)進行網路連線的行為特徵分析；偵測代理人會將收集到的傳輸流進行協定分類，當偵測代理人分析得到結果後，會將結果通知給 SDN 控制器。最後當 SDN 控制器收到偵測代理人的通知後會根據 P2P 的類型透過 OpenFlow 協定對 OpenFlow 交換器中的流向規則表(Flow Table Rules)做對應的修改，藉此管理整個網路的連線行為，達到一個自動化的網路流量控管服務。

圖九 系統運作流程架構



資料來源：本研究繪製

二、運作流程

系統大致分為四個步驟，系統運作的流程架構如圖九，分別如下：

(一)傳輸流與特徵抽出器(Traffic Flow & Feature Extractor)

為了要得到一些可用的資訊來分類不同主機之間的網路運輸(Network Traffic)，封包在鏡像到代理人後將在此流程重新整理，並基於以下五項因素組合成一筆運輸資料流：Source IP、Source Port、Destination IP、Destination port、Protocol，最後將資料流匯集並從中計算出多種特徵集合。

(二)P2P 應用偵測模組(P2P Application Detection Module)

機器學習裡面很重要的一環就是特徵選擇，特徵選擇的好壞可以左右識別的準確度，在一些殭屍網路偵測的研究中[14][15]，適合的特徵如下：

1.封包計數(Packet Count)。

[14] P. Narang, J. Reddy, and C. Hota, "Feature Selection for Detection of Peer-to-Peer Botnet Traffic," In Proceedings of the 6th ACM India Computing Convention, 2013.
[15] P. Narang, S. Ray, C. Hota, and V. Venkatakirshnan, "PeerShark: Detecting Peer-to-Peer Botnets by Tracking Conversations," In EURASIP Journal on Information Security, oct, 2014.

2.封包大小(最小、最大、平均及標準差)(Packet size (Min, Max, Mean and Standard Deviation))。

3. 總量(流的大小)(Total Volume (Flow Size))。

4.間隔時間(最小、最大、平均及標準差)(Inter-arrival Times (Min, Max, Mean and Standard Deviation))。

5. TCP 推送旗標計數(TCP Push Flag Count)。

6.流持續時間(Duration of the Flow)。

7.標頭使用總位元組數(Total Bytes Used for Headers)。

8. TCP 緊急旗標計數(TCP Urgent Flag Count)。

將此特徵作為機器學習演算法，諸如 KNN(K-nearest Neighbors)、SVM (Support Vector Maching)與 Random Forest 的參數以提高辨識率。

(三)回報 RYU 控制器(Report to RYU Controller)

RYU[16]為 SDN 控制器規範中的一個實作框架，在此以 RYU 作為本研究 SDN 網路架構中的交換器，因此偵測代理人分析出網路架構中某個交換器底下的某筆連線行為符合正在監測的 P2P 應用後，就會通知 RYU 控制器依照應用的種類管理這個連線，例如監測到殭屍網路行為便丟掉封包等。

(四)OpenFlow 交換器流向規則修改(Flow Rule Modify on OpenFlow Switch)

當 RYU 控制器收到偵測代理人的通知後，會透過 OpenFlow 協定跟交換器溝通修改流向規則，而交換器則會根據這些規則來控制經過它的封包，達到自動化的網路行為控管服務。

三、實驗設計與結果

本實驗參考論文 PeerRush[17] 的做法，使用 Botnet-zeus, Skype, Utorrent, eMule 等四種 P2P 程式的流量資料進行實驗，實驗中將殭屍網路 Botnet-zeus 當作一類，其他的三種 P2P 應用程式當成另一類。

(一)K 值實驗

本實驗使用 Scikit-learn[18]的最近鄰居(K-Nearest Neighbor Learning, KNN)演算法以及交叉驗證(Cross-Validation)來訓練分類器，以及測試這些分類器的效果，由於不同的 K 值設定將會影響到分類的準確率。必須先針對各種 K 值進行實驗以取得最佳成果，表二為使用 KNN 演算法選取不同的 K 值訓練出來的模型

[16] "A Component-based Software Defined Networking Framework," http://osrg.github.io/ryu/, 2015/12/16.
[17] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "PeerRush:Mining for Unwanted P2P Traffic," In Detection of Intrusions and Malware, and Vulnerability Assessment, volume 7967 of Lecture Notes in Computer Science, p.62-82. Spriger Berlin Heidelberg, 2013.
[18] Scikit-learn, "scikit-learn, Machine Learning in Python," http://scikit-learn/org, 2016/1/4.

及測試結果，表中第一欄為選定的 K 值，第二欄是該 K 值所訓練出來分類器的判斷準確率，第三欄及第四欄分別是接收者操作特徵曲線(Receiver Operating Characteristic Curve, ROC)[19]及精確率-召回率曲線(Precision-Recall Curve, PR)[20]下的面積(Area Under Curve, AUC)，其面積的值愈接近 1 表示分類效果愈好，從表二來看，當 K 值大於 3 時 ROC 曲線及 PR 曲線下的效果相去不遠，在此選擇 K＝7 進行以下實驗。

表二 最近鄰居分類器中不同 K 值之準確率

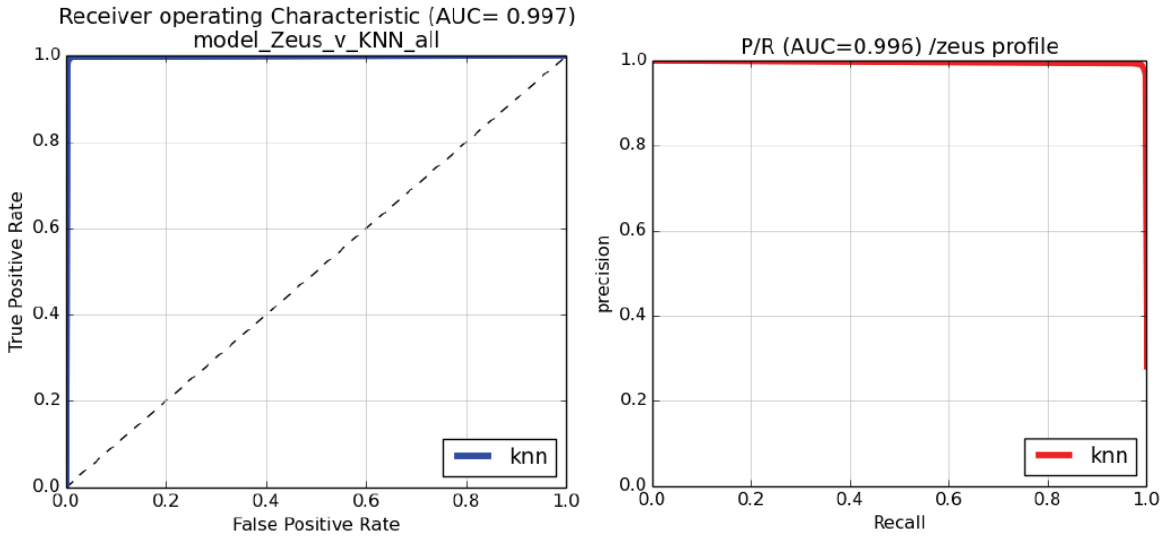
K 值	準確率	ROC 曲線下面積	PR 曲線下面積
1	0.9943	0.9945	0.9923
2	0.9945	0.9957	0.9940
3	0.9948	0.9966	0.9950
4	0.9951	0.9967	0.9950
5	0.9946	0.9970	0.9952
6	0.9943	0.9972	0.9955
7	0,9943	0,9976	0.9961
8	0.9945	0.9976	0.9957
9	0.9940	0.9978	0.9965

資料來源：本研究繪製

(二)特徵選取實驗

實驗一開始先用所有的特徵來訓練分類器，其訓練結果如圖十，圖中可以看到訓練結果相當傑出，表示 Botnet-zeus 與其他 P2P 應用程式的行為有很大的不同。

圖十 使用所有特徵產生的訓練結果



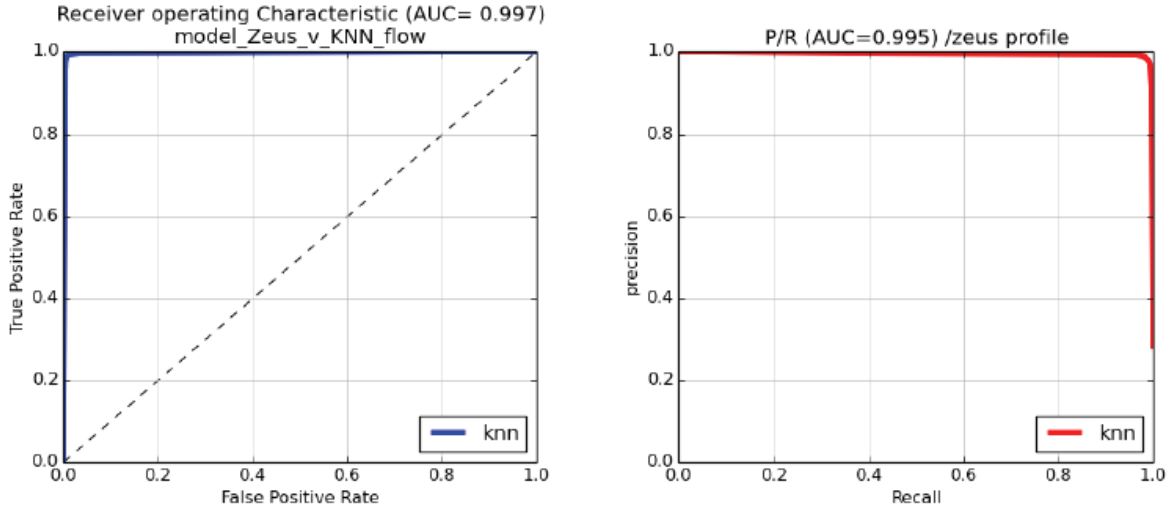
[19]〈ROC 曲線〉《維基百科》，http://zh.wikipedia.org/ROC 曲線，2016 年 1 月 4 日。
[20] WikiPedia, “Precision and Recall,” http://en.wikipedia.org/wiki/Precision_and_recall, 2016/1/4.

附註：
1. 接收者操作特徵(Receiver Operating Characteristic)。
2. 曲線下面積 AUC(Area Under the Curve)。
3. 使用所有特徵訓練 K-近鄰算法模組(Model_Zeus_v_KNN_all(k-nearest Neighbors))。
4. 真陽性率(True Positive Rate) (左圖 Y 軸)。
5. 偽陽性率(False Positive Rate) (左圖 X 軸)。
6. Zeus 設定檔(Zeus Profile)。
7. 精確率(Precision) (右圖 Y 軸)。
8. 召回率(Recall)(右圖 X 軸)。

資料來源：本研究繪製

接著使用 Flow 相關的特徵，包含去回的流量與封包數來進行分類器訓練與測試，其結果如圖十一，從實驗結果可以發現單用 Flow 相關的特徵就能明顯地將 Botnet-zeus 與其他的 P2P 應用程式區分開來，因為殭屍網路在溝通時會為了避免暴露行為而盡量減少傳輸流量，所以很容易與一般的 P2P 應用程式區別。

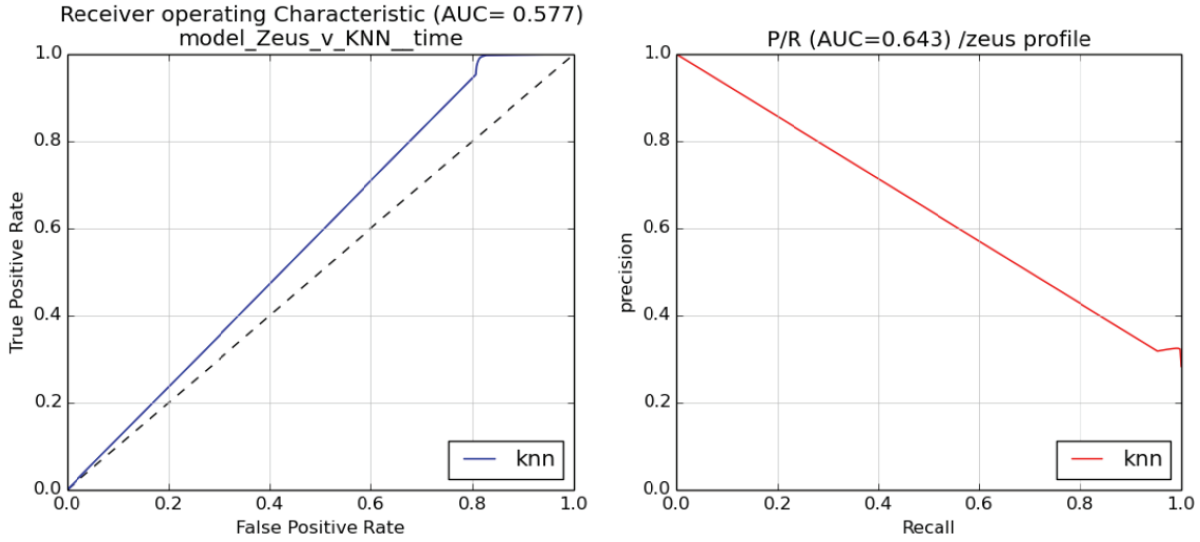
圖十一 Flow 相關特徵曲線圖



資料來源：本研究繪製

值得注意的是，當使用時間間隔(Inter Arrival Time)相關特徵來訓練與測試分類器時，包含的特徵值有去回兩種封包之間的間隔時間最大、最小、平均及標準差，其測試結果如圖十二。可以發現訓練出來的分類器效果並不好，說明 Botnet-zeus 在溝通頻率與一般的 P2P 應用程式差異並不大，只有在封包大小上不一樣而已。

圖十二 時間間隔相關特徵曲線圖



資料來源：本研究繪製

(三)實驗結果

在各種特徵的正確率與曲線覆蓋面積的比較下，結果如表三，我們可以發現 Flow 與封包相關的特徵可以有效地分類 Botnet-zeus 與其他 P2P 應用程式的行為，說明雖然殭屍網路會想要試著隱藏它們的溝通行為，但只是透過傳送較小的封包來溝通，而封包傳送的頻率並沒有比較低。

表三 各種特徵正確率與曲線覆蓋面積比較表

特徵種類	準確率	ROC 曲線下面積	PR 曲線下面積
流(Flow)	0.9916	0.9967	0.9944
封包(Packet)	0.9940	0.9979	0.9976
採樣流(Sflow)	0.9738	0.9903	0.9813
間隔時間 (Inter Arrival Time)	0.4156	0.5795	0.6301
活動空閒時間 (Active Idle Time)	0.7913	0.8712	0.7285
標頭(Header)	0.5027	0.7117	0.6224

資料來源：本研究繪製

結論

雖然這次的實驗顯示出 Botnet-zeus 與一般的 P2P 應用程式在封包大小與傳輸流量上有明顯的差別，但在其他的特徵上並無明顯差異，隨著時間與技術的演進，殭屍網路的溝通模式也將會隨著攻擊者的調整而無法只用這些特徵來識別它們，甚至可能針對目前選擇的偵測方法去設計新的溝通模式，所以資安人

員仍必須要持續關注殭屍網路的改變與適時的更新分類的方法，才能有效的阻止殭屍網路。

軟體定義網路的出現，讓網路封包的管理更具彈性、有效率也更即時，透過 OpenFlow 的架構，可將網路封包隨時導向任意建立的分析器，一旦偵測到異常狀態便能立刻反應處理，更能配合機器學習等各種演算法加強偵測、防禦能力。因此，本研究針對殭屍網路所發動的分散式阻斷服務攻擊，利用軟體定義網路架構結合機器學習演算法，提高分散式阻斷服務攻擊偵測率，進一步達到防禦效果，未來更可以持續擴充偵測防禦功能，降低甚至避免國軍網路遭到網路攻擊之傷害。

參考文獻

一、J. Leonard, S. Xu, and R. Sandhu, “A Framework for Understanding Botnets,” In Proceedings of the International Workshop on Advances in Information Security, Fukuoka Institute of Technology, 2009.

二、S. S. Silva, R. M. Silva, R. C. Pinto, R. M. Salles, “Botnets: A survey,” In Computer Networks, vol. 57, no.2, 2013.

三、M. Stevanovic, and J. M. Pedersen, “An efficient flow-based botnet detection using supervised machine learning,” In Computing, Networking and Communications (ICNC), 2012 International Conference on, Feb 2014.

四、S. C. Guntuku, P. Narang, and C. Hota, “Real-time Peer-to-Peer Botnet Detection Framework based on Bayesian Regularized Neural Network,” In CoRR, 2013, arXiv:1307.7464v1

五、M. Stevanovic and J. M. Pedersen, “Machine learning for identifying botnet network traffic,” Aalborg University, Technical Report, 2013.

六、L. Bilge, D. Balzarotti, W. K. Robertson, E. Kirda, and C. Kruegel, ”Disclosure: detecting botnet command and control servers through large-scale netflow analysis,” In ACSAC, 2012.

七、P. Phaal, ”sFlow Specification Version 5,” July 2004.

八、S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, ”Detecting P2P botnets through network behavior analysis and machine learning,” In Proceedings of 9th annual conference on privacy, security and trust (PST’11), 2011.

九、M. Stevanovic, and J. M. Pedersen, ”An efficient flow-based botnet detection

using supervised machine learning," In Computing, Networking and Communications (ICNC), 2012 International Conference on, Feb 2014.

十、OpenFlow: Enabling Innovation in Campus Networls, http://www.openflow.org/documents/openflow-wp-latest.pdf

十一、D. Kreutz, F. M. V. Ramos, and P. Verissimo, " Towards Secure and Dependable Software-Defined Networks," In Proceedings of the second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking (HotSDN), 2013.

十二、N. McKeown, et al. "OpenFlow: Enabling Innovation in Campus Networks," In SIGCOMM Computer Communications Review, vol. 38, no.2, 2008.

十三、P. Narang, J. Reddy, and C. Hota, "Feature Selection for Detection of Peer-to-Peer Botnet Traffic," In Proceedings of the 6th ACM India Computing Convention, 2013.

十四、P. Narang, S. Ray, C. Hota, and V. Venkatakirshnan, "PeerShark: Detecting Peer-to-Peer Botnets by Tracking Conversations," In EURASIP Journal on Information Security, oct, 2014.

十五、"A Component-based Software Defined Networking Framework," http://osrg.github.io/ryu/.

十六、B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "PeerRush:Mining for Unwanted P2P Traffic," In Detection of Intrusions and Malware, and Vulnerability Assessment, volume 7967 of Lecture Notes in Computer Science, pages 62-82. Spriger Berlin Heidelberg, 2013.

十七、"scikit-learn, Machine Learning in Python," http://scikit-learn/org.

十八、"Receiver operating characteristic," http://en.wikipedia.org/Receiver_operating_ characteristic.

十九、"Precision and Recall," http://en.wikipedia.org/wiki/Precision_and_recall.