以信譽機制為基礎之巨量資料偵測可疑網址

陳嘉玫、黃哲諄、歐雅惠
國立中山大學資管所
cmchen@mis.nsysu.edu.tw

摘要

由於網際網路的便利性，傳統服務與產業漸漸轉向網路型的服務與行銷，促使人們生活必需與網路息息相關，網路的資料量也與日俱增。大數據領域權威專家-麥爾荀伯格（Viktor Mayer-Schonberger）認為大數據(Big data)時代來臨，不論是在商業、資訊科技、公共行政、教育、醫療等領域中，誰能在龐雜的訊息中掌握關鍵，誰就能取得解決問題的先機[1]。因此，在大量網路資料中，找到關鍵的威脅訊息，進而掌握問題，解決問題，是網路安全領域越來越顯著的趨勢。也因為經濟效益的誘惑，駭客更是對利用網路犯罪取得利益樂此不疲，研究出更多更新穎的攻擊模式，路過式下載攻擊(drive-by download) 就是近來被作為攻擊的模式之一。路過式下載攻擊就是未經電腦使用者的允許，當使用者瀏覽網頁時即自動下載惡意程式的攻擊方式。利用更多的混淆技術，反向連結方式和藉由HTTP協定的特性,使得傳統的入侵偵測系統與防火牆難以偵測。因此，本研究提出利用監聽(sniffer)的方式，對 HTTP 流量進行動態檢定。

多數被瀏覽的網站都屬於良性的網站，因此，本研究使用信譽系統(reputation system)先行過濾掉良性網站，找出可疑的網站，再透過誘捕系統進行偵測並分析，如此一來可以減輕誘捕系統的負載量，可以更有效能的分析攻擊的特徵。在真實實驗環境中，我們甚至可以透過信譽系統的過濾，減少誘捕系統的分析時間，並可從中偵測到使用者瀏覽路過式下載攻擊的紀錄。

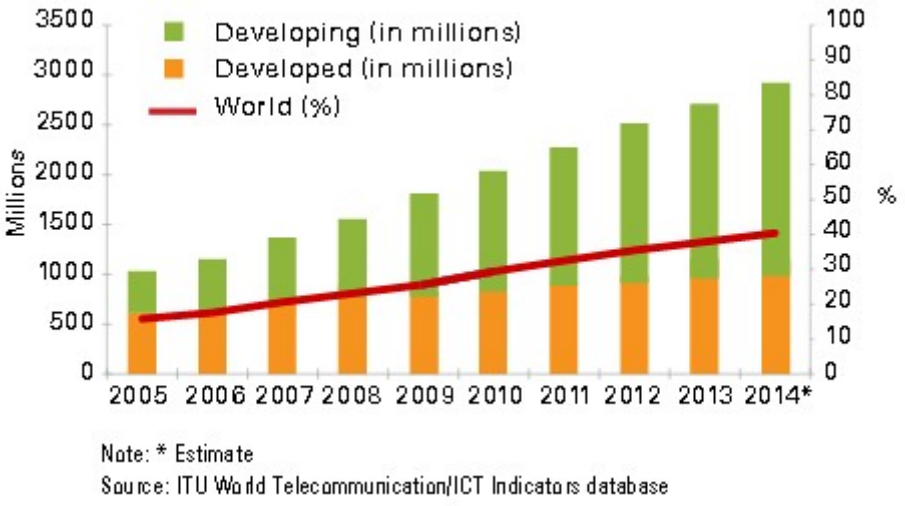
本研究的信譽系統不需要使用 WHOIS 的資料庫，適用於各種網域名稱，我們使用 3 個特徵集合共計 12 個特徵值，並基於機器學習建立分類模型，其正確分類的成功率達 90.9%。

關鍵詞：路過式下載攻擊，客戶端誘捕系統，信譽系統

壹、介紹

根據聯合國 2011 年的資料指出，全球上網人口較 2010 年成長了 408%[2]，大量的使用者帶來龐大的商機，軟體開發人員更是利用低成本開發與社群網站的契機，大量撰寫網頁應用程式吸引使用者下載使用。因此引來駭客覬覦這類型的經濟效益，利用應用程式的安全漏洞進行攻擊並竊取使用者的個人資料。根據 Cenzic[3]研究報告指出 2010 年的網路與基礎設備弱點的 57%是屬於 Web 弱點，主要原因是駭客可以輕易入侵並感染成為惡意網站。隨著資訊通信技術的蓬勃發展與移動式裝置的興起，網路使用率增加

速度更快，如圖一，根據 ITU (International Telecommunication Union)統計，2014 年底預估全球網路普及率將達到 40% ，從 2011 年的 20 億使用人口到 2014 年高達 30 億使用人口[4]，龐大的網路資料量也應運而生，也增加了網路威脅偵測的困難度。但是，根據 Google 使用 crawler 調查發現在 google 的搜尋結果中約只有 1.3%的網站是惡意網站，且發現這些惡意網站多數會發動路過式下載攻擊。因此，本研究試圖先將大量的良性網站過濾掉，減少分析的資料量，將剩下少數的惡意網站進行再透過誘捕系統進行測試。



圖一：全球網路普及率[4]

路過式下載攻擊通常會使用 script 語言(javascript, vbscript, actionscript)將 shellcode 載入記憶體，並刺探(exploit)客戶端元件的弱點。其散播過程分成 3 個階段[5]:瀏覽受感染的惡意網站，重新導向駭客設定的網站，下載並安裝惡意程式。因為這類的攻擊模式缺乏周期性的特性，使得利用時間序列作為偵測基礎的 IDS 難以正確偵測。加上可以利用 script 的 substring、fromcharcode、eval 等函式隱藏 shellcode，因此以 signature 作為偵測基礎的 IDS 也很難偵測其異常。

客戶端誘捕系統(Client Honeypot)是一種有效防禦路過式下載攻擊的網站安全架構，主要是利用具有弱點的瀏覽器刻意與網站互動，引誘駭客發動攻擊，一旦客戶端誘捕系統的機碼(registry)、檔案或是程序遭到竄改，即可判斷該系統受到路過式下載攻擊。因為多數研究以 crawler 的方式操作客戶端誘捕系統，在網際網路循序的進行搜尋，逐筆的搜尋延緩進行分析的時效性，故增進搜尋惡意網站的效率成為該系統急待解決的問題。根據 Honeymonkey[7]的相關研究指出要讓客戶端誘捕系統下載並安裝惡意程式的過程平均需要 30 秒的時間，冗長的判斷時間相對的會影響其效能。

在成千上萬的網路搜尋資料中，雖然有部分是惡意的網站，但是絕大部分仍然屬於良性網站，誘捕系統不應該把資源浪費在判斷良性網站上。因此，本研究使用信譽系統

先行過濾良性網站，再將剩下的可疑網站進行偵測並分析。

本研究主要具有 5 大特色：

(1) 利用監聽方式直接觀察使用者的存取狀態，相較 crawler 方式的研究更能直接度量路過式下載攻擊的危害程度。

(2) 在 client hoenypot 系統利用動態檢定(execution-based)進行檢測，較能輕易檢測出含有混淆技術的惡意網頁。

(3) 使用信譽系統重新配置誘捕系統的運算資源，大幅提高誘捕系統進行檢測大量資料時的效率。

(4) 信譽系統不需使用 WHOIS 資料庫，可以更容易處理普遍的網域名稱。

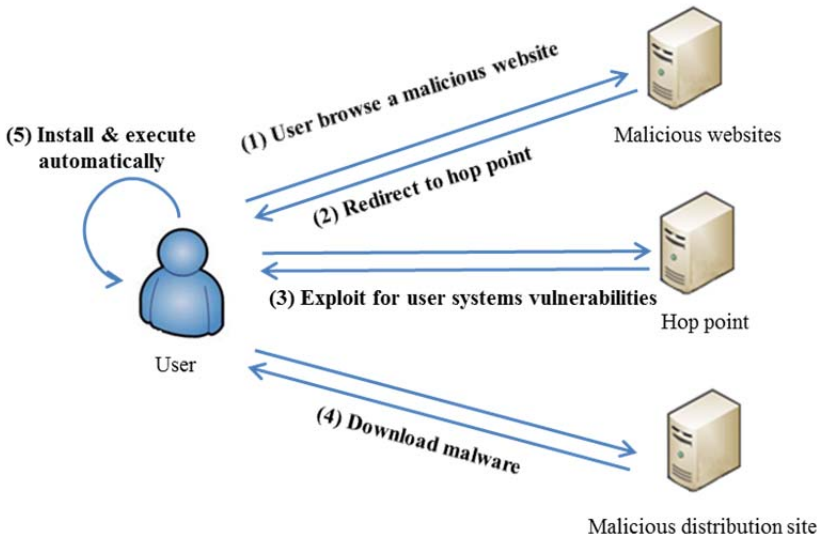
(5) 不僅使用 DNS 的 A-Tape 資料，也使用 NS-Tape 資料，取出更好的特徵。

貳、相關研究

網際網路的迅速發展為人們提供了一個更便利的溝通與服務方式，隨著 Web 2.0 的使用量增加，要求瀏覽網站的安全性也變得更加重要。大多數駭客會利用瀏覽器的組件，發動攻擊，如 JavaScript、Flash、widgets 等元件。瀏覽器可以透過這些組件很容易達到重新導向的作用，但傳統的 server honeypot 不能作到檢查和跟踪攻擊來源的功能。因此，Spitzner[6]提出了解決這些問題的 client honeypot 的方法。根據 Spitzner 觀察，傳統的 server honeypot 是不能檢測針對瀏覽器的威脅，而 client honeypot 主要是在收集有關客戶端攻擊，駭客在惡意網站中植入啟動路過式下載攻擊的惡意程式，當使用者瀏覽網頁時便自動啟動惡意程式。

一、路過式下載(drive-by download)

又稱為偷渡式下載，其說明如圖二，當使用者瀏覽網頁時不經意點擊具有惡意 HTTP 程式碼的網頁時，會被導到一個跳板網頁並被試探系統的弱點，然後將惡意程式進行下載到系統中，即使合法的網站也可能被進行滲透。通常是利用 Script 可以溝通互動的特性或是 plug-in 弱點。



圖二：Drive-by download 過程

Jim 提出一個 BEEP(Browser-Enforced Embedded Policies)機制用以防止 JavaScript injection 攻擊[7]，對於每個網頁定義其安全策略，使其在執行 script 時多一層保護。但是，要找到各種網頁都適合的安全機制不是件容易的工作。Hallaraker 和 Vigna[8]提出一個將 JavaScript 解譯內容進行審查的系統，對 JavaScript code 執行並比對其特徵。

二、偵測惡意網站

Seifert 等提出了一種用於檢測具有高互動惡意網站的 client honeypot[9]，當使用者瀏覽網站時可以作分類和監測狀態的變化。作者利用 J4.8 演算法找出惡意屬性，以評估其預測的能力。當狀態改變時，該網站會被歸類為惡意網站。

Sadan 和 Schwartz 提出了一種新的方法來預測網站的安全性[10]，其分析的參數是以社群網路為主。是根據計算 URL 出現的數量和質量的聲譽作為決定該網站是否為惡意網站的標準。作者使用 WhiteScript 演算法評估和過濾內容作為聲譽指標。Lin 等人結合語意分析模型對惡意程式進行檢測與分類[11]，被模糊化的網頁可以藉由語義的方法進行鑑定，作為瀏覽網頁的安全機制。這種模式結合多種防禦技術與匹配多種惡意 script，可以藉以加強監測和分析能力。

Antonakakis 等 [12]推薦一個動態的信譽系統，是利用 DNS 計算分數，以評估新的或未知網域的聲譽。該系統使用被動式 DNS 查詢數據和分析網域的特徵值作為建構已知的合法網域和惡意網域的基礎。

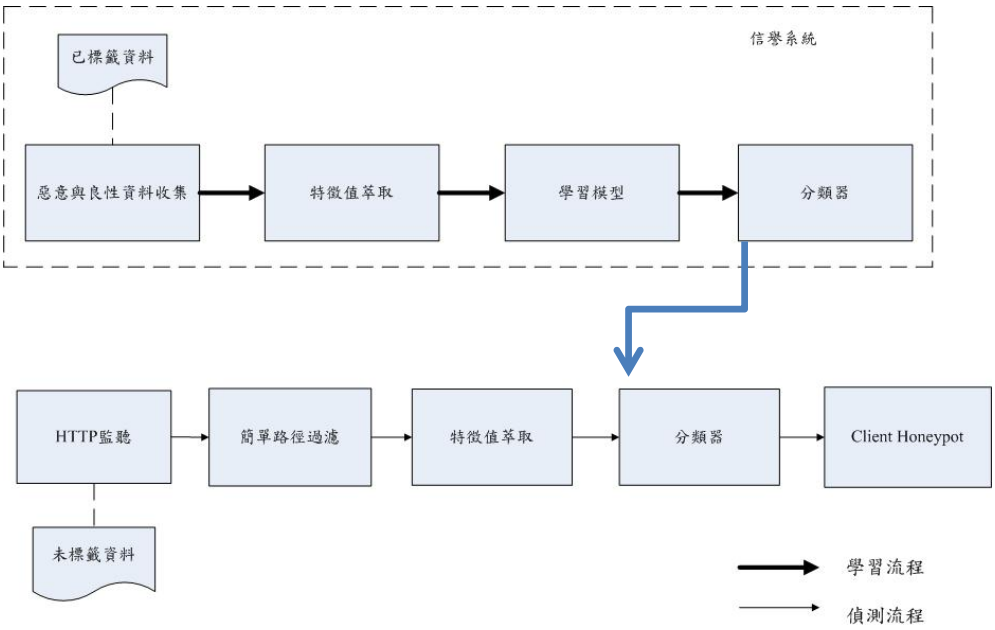
參、研究方法

本研究的主要目的是提供檢測路過式下載攻擊和發現惡意網站的簡單而準確的方法。比較以 signature 為基礎的方法，透過通過 execution 為基礎的方法更容易檢測零日攻擊(Zero day attack)和混淆技術。在檢測過程中，越來越多的研究，使用 client honeypot 做為檢測惡意網站的方法。然而，它存在一個嚴重的缺點就是偵測的時效性，通常要感染路過式下載攻擊並恢復是需要很長一段時間。

本研究採用的信譽系統，透過機器學習演算法先行刪除大量的良性網站，並重新分配 client honeypot 的運算資源。

一、研究架構

為了發展一套有效的信譽系統進行過濾大量的良性網頁，本研究將信譽系統的架構分成兩個階段包含學習流程與偵測流程，如圖三所示，其設計包括七個主要元件：資料收集元件，路徑過濾元件，特徵萃取元件，學習模型，分類模型，HTTP 監控元件，與 client honeypot。學習流程是用來建立信譽系統；檢測流程被使用來監聽流量以檢測路過式下載攻擊。



圖三：系統架構

系統架構的部分流程描述如下:

(A) 資料收集元件的基礎是從幾家知名黑名單網站和作惡意網站排名的網站收集到，像是從 clean mx[13]，malwaredomainlist[14]和 phishtank[15]收集到惡意網站，良性的網站則是從 Alexa[16]，Open Directory Project[17]收集而來。

(B) 路徑過濾元件：路過式下載攻擊的攻擊流程分成三個步驟:瀏覽一般網站、重新導向跳躍站台與下載並執行惡意程式。路徑數量是一個重要的特點，以確定是否存在攻擊整合包(toolkit)的特性。然而，許多良性網站也有相同的特性，所以我們需要建立一個信譽系統再進行分析。

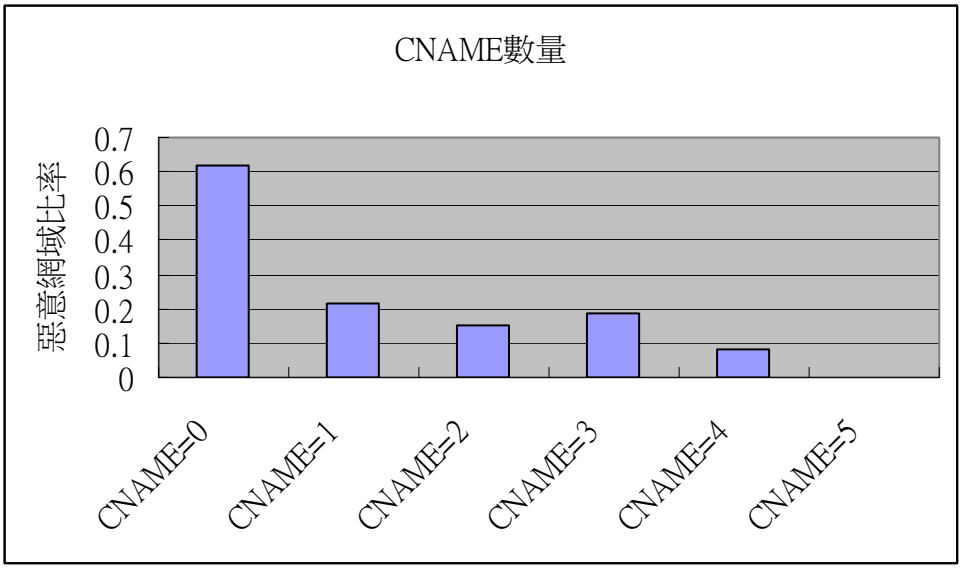
(C) 特徵萃取元件:透過 dig 指令取得 DNS 紀錄，用來作為 DNS 的查詢工具，取得網域相關資訊。有效地消除無意義的部分，以達到良好的分類功能。我們計算出分類和學習的特徵值模型。結果可以分為三個部分包括 answer section、authority section 與 additional section 並歸納出 12 個特徵如表一。

表一：特徵分類

特徵集	#	特徵名稱
Answer-Based Features	F_1	CNAME 數量
	F_2	IP 數量
	F_3	Numeric ratio
	F_4	LMS
	F_5	Leave Domain Type
Authority-based Features	F_6	NS 數量
	F_7	Rogue
ASN-based Features	F_8	ASN 數量
	F_9	國家數量
	F_{10}	Centralize
	F_{11}	Mode Equivalence
	F_{12}	對應到的國家

針對部分特徵的詳細描述如下:

a. CNAME(canonical name record)的數量:駭客通常不要求服務質量，因此，存在 CNAME 的紀錄就代表是惡意網站的機會就減少了。



圖四：CNAME 數量跟惡意網域的比率關係

b. Leave Domain Type 的含意:因為 ccTLD 與 gTLD 的命名規則較嚴格，許多商業組織需要取得更上層的網域名稱，故出現了 DLD(delegation level domain)，就不需受限於 ccTLD 與 gTLD 的命名規則。但是 DLD 會出現含糊的字眼，容易被惡意網站誤用，也容易誤導使用者。

c. 有意義的字串長度（LMS）：計算網域名稱中有意義的字串佔全部字元的比例與數字所佔的比例。透過自動化註冊的網域名稱的存活時間（expiration time）較一般正常的短，並且註冊的網域名稱通常沒有什麼意義。因此，計算網址中無意義的字元也可以確定是否為惡意程式。其計算方程式如下:

$$LMS = \frac{Max(\bigcup_{j=1}^{N} \frac{\sum_{i=2}^{M} p(c_{i-1}, c_i)}{M-1})}{M} \quad (1)$$

d. NS 數量:網域名稱的結構中，每個 token 是由[.]連結而成，若 level domain type 是 gTLD 或 DLD 則取 2LD 判斷，若 level domain type 是 ccTLD 則取 3LD 作為判斷惡意網站的標準，以利下一步進行分群。

e. Rogue:不管是良性網站或是惡意網站都有集中在某些網域的特性。因此本研究利用 Density-based 演算法(DBSCAN)將網域分群，並計算每一分群中惡意網站所佔的比例來判斷是否該分群是否為惡意網站的網域集合。計算方程式如下:

$$Rogue = \frac{Number_{malicious}}{Number_{total}} \quad (2)$$

如果 Rogue>0.5，就可以判斷該分群為惡意網站集合的機率較大，落在該區域內的網站是惡意的機率也較大。

f. ASN 的數量:因為 fast-flux domain 的 IP 多為分散各地的受害主機，觀察同一個 ASN 對應的 IP 是否屬於同一個網域作為判斷標準。

二、建立模型

本研究利用機器學習的方法建構信譽系統，首先從數家知名的黑名單網站像是 cleanmx[13]，malwaredomainlist[14]和 phishtank[15]收集到 14931 個惡意網站；而 63225 個良性網站則是從 Alexa[16]，Open Directory Project[17]收集而來。再利用 J48 演算法作為分類的模型。以 information gain 挑選最佳特徵作為決策樹的節點，其計算方程式如下:

$$IG(E,f)=entropy(E)-entropy(E|f)=entropy(E)-\sum_{v\in vaules(f)}\frac{|V|}{|E|}\times entropy(V) \qquad (3)$$

其中的

$$V=\{x\in E\,|\,vaule(x,f)=v\} \qquad (4)$$

肆、效能評估

本研究使用信譽系統改善 client honeypot 效能不彰的問題，並在真實環境中偵測路過式下載攻擊的情況。首先將已標記的資料進行驗證，度量信譽系統的精確率與召回率。再者，取 URL 監聽的資料，與 divide and conquer 的研究比較效率。最後，在真實實驗環境中進行概念驗證，觀察路過式下載攻擊。

一、度量信譽系統

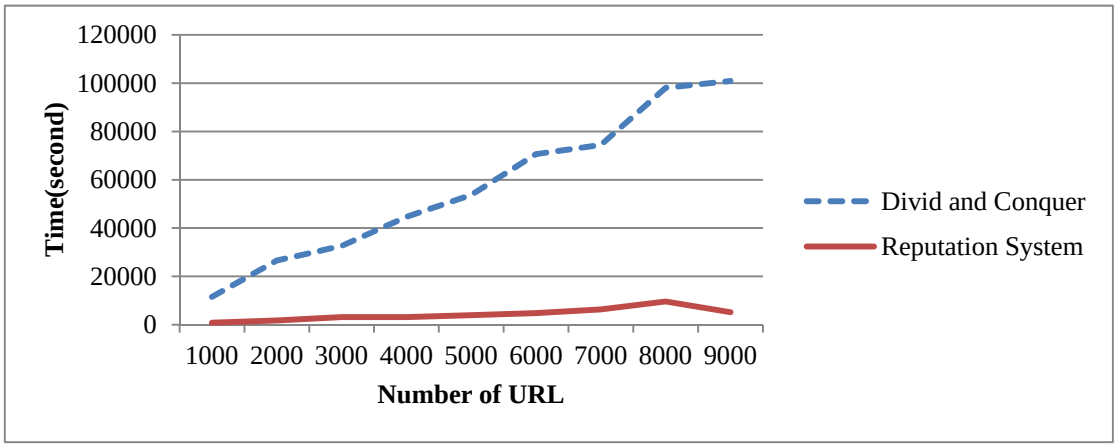
驗證信譽系統的精確率，本研究採用 10-fold cross validation 的方式作交叉驗證。並進一步探討個別 Feature 與 Feature Set 對分類的影響。，其分類結果如下。

表二：信譽系統分類結果

預測 / 實際	良性網站	惡意網站
良性網站	8482	518
惡意網站	1120	7880
正確率(true positive rate)	87.6%	
誤判率(false positive rate)	5.8%	
精確率(precision)	93.8%	
召回率(Recall)	87.6%	

二、相較 divide and conquer 的效率

為了解決 client hoenypot 效能不彰的問題，Seifert 在 2008 年提出以 divide and conquer 演算法改善其效能。在 URL 的資料量小的時候確實具有絕佳的效果，但是並不適用資料量較大的 HTTP 監聽狀況。本研究先利用信譽系統過濾去除良性網站，使得 client honeypot 的運算達到更好的效能。其結果如下:



圖五：相較 divide and conquer 的效率

三、真實實驗環境

本研究在 10 個 class C 的網路環境中進行概念驗證(proof of concept)，並真實偵測到未知的路過式下載攻擊。由於本研究使用 HTTP 監聽方式取得資料，更接近實際使用狀況。也確實更迅速能執行判斷與分析的過程。其結果如下:

表三：信譽系統執行所需的時間

第 n 天	n=1	n=2	n=3	n=4
執行時間	21:16:20	20:26:04	17:02:11	16:08:28
第 n 天	n=5	n=6	n=7	
執行時間	5:42:23	6:55:23	8:35:46	

平均每天花費約 14 小時，處理當天的資料量。並發現真實的攻擊實例。透過查詢多個紀錄惡意網站的知名網站，都沒有紀錄該網站為惡意網站。更證實本研究的信譽系統可以更迅速偵測到路過式下載的攻擊。

伍、結論與未來工作

Client honeypot 是有效偵測路過式下載攻擊的系統，但是存在效能不彰的問題，多數研究也都針對這個問題相繼提出改善的工作。本研究利用信譽系統，有效配置 client honeypot 的運算資源顯著優於只使用 divide and conquer 的效能。加上本研究不需要使用 WHOIS 資料庫的資料，讓信譽系統可以適用各種網域名稱，優於其他的評分系統。利用 HTTP 監聽方式，更能確實反映真實的網路環境使用情況。

本研究中，信譽系統處理的 URL 資料每筆都視為獨立個體；但實際情況，URL 之間必定存在相依的關係。在未來研究上將更進一步將 URL 的相依性作分析，以提升信譽系統的準確度。在即時通報方面，相信可以在更進一步改善其時效性。

參考文獻

[1] 林士蕙，”未來 10 年，賣資料比賣硬體賺錢”遠見雜誌，2014(4)，vol.334，http://www.inside.com.tw/2014/05/23/mayer-schonberger-interview

[2] Internet World Stats, “Usage and Population Statistics,” http://www.internetworldstats.com/stats.htm.

[3] CENZIC, “Web Application Security Trends Report,” http://www.cenzic.com/downloads/Cenzic_AppSecTrends_Q3-Q4-2010.pdf

[4] Wikipedia, “ Global Internet usage”, http://en.wikipedia.org/wiki/Global_Internet_usage

[5] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose, “All Your iFRAMEs Point to Us,” *Proceedings of the 17th conference on Security symposium*, 2008.

[6] L. Spitzner, “Honeypots：Tracking Hackers,” Addison Wesley, 2002.

[7] T. Jim, N. Swamy, and M. Hicks, “ Defeating script injection attacks with browser-enforced embedded policies,” Proceedings of the International World Wide Web conference, 2007.

[8] O. Hallaraker and G. Vigna, "Detecting malicious JavaScript code in mozilla," Proceedings of the 10th IEEE International Conference on Information, Communications and Signal Processing (ICECCS 2005), pp. 85-94, 2005.

[9] C. Seifert, I. Welch and P. Komisarczuk, “Identification of Malicious Web Pages Through Analysis of Underlying DNS and Web Server Relationships”, 33rd Annual IEEE Conference on Local Computer Networks, 2008.

[10] Z. Sadan and D. G. Schwartz, “WhiteScript：Using social network analysis parameters to balance between browser usability and malware exposure,” Computers & Security, Vol. 30, No. 1, pp.4-12, 2010.

[11] S.F. Lin, Y.T. Hou, and C.M. Chen, B.C. Jeng, and C.S. Laih, "Malicious Webpage Detection by Semantics-Aware Reasoning," in Proceedings of The International Conference on Intelligent Systems Design and Applications, pp. 115-120, 2008.
[12] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, “Building a Dynamic Reputation System for DNS,” Proc. USENIX Security Symposium, 2010.
[13] CLEAN MX, “CLEAN MX realtime database,” http://support.clean-mx.de/clean-mx/viruses.
[14] Malware Domain List, “Malware Domain List,” http://www.malwaredomainlist.com/.
[15] Phishtank, “Phishtank,” http://www.phishtank. com/.
[16] Alexa, “Alexa the Web Information Company,” http://www.alexa.com/.
[17] Dmoz, “Open Directory Project,” http://www.dmoz.org/.