行政院國家科學委員會專題研究計畫 成果報告

前瞻性雲端安全儲存、防護、行為分析與觀測平台--子計畫二：基於機器碼之 Windows 惡意程式行為分析雲端平台 (2/2)

研究成果報告(完整版)

計 畫 類 別 ： 整合型
計 畫 編 號 ： NSC 100-2218-E-009-004-
執 行 期 間 ： 100 年 08 月 01 日至 101 年 07 月 31 日
執 行 單 位 ： 國立交通大學資訊工程學系（所）

計 畫 主 持 人 ： 謝續平

計畫參與人員： 博士班研究生-兼任助理人員：卓政逸
博士班研究生-兼任助理人員：陳仲寬
博士班研究生-兼任助理人員：李秉翰

公 開 資 訊 ： 本計畫可公開查詢

中 華 民 國 101 年 10 月 31 日

中 文 摘 要 ： 全系統層次的惡意程式行為分析技術可以比以往傳統靜態分析技術更精確地判斷一個執行檔案行為是否影響系統安全。本計畫擬提供一個可信賴性的分析服務與雲端應用層的安全防禦做結合，藉此為雲端服務技術做安全性的把關。本研究將探討此技術如何結合虛擬機器技術與雲端運算，以適用於分析工作的分派，使檔案可以自動地、大量地透過連續的分析檢驗來達到程式行為分析。有別於傳統的軟體即服務（SaaS）模式，本計畫除了提供可疑檔案上傳平台介面外，還開發出一個較為主動的檢驗服務，針對網際網路上現有檔案進行蒐集與分析，除了可供使用者判斷下載檔案的安全性，也可讓資安人員用來驗證其開發出的檢測軟體是否做出正確的判斷。

本研究計畫共有兩大研究成果：（一）使用全系統層次的行為分析技術來擷取惡意程式行為以達到更準確的分析、（二）開發與設計雲端架構分析平台以提高安全分析產能。雲端架構分析平台可以自動地蒐集網路上的惡意程式樣本進行分析，並且對其行為的分析結果利用分散式資料庫儲存。而分析的階段皆以自動化、平行化的方式來增加運算節點的使用率以提高分析的產能。本計畫將會實際設計與開發一個雲端架構分析平台，並會對惡意樣本做行為分析，以利用其惡意行為來偵測未知的惡意程式樣本。

中文關鍵詞： 惡意程式、程式行為分析、雲端安全

英 文 摘 要 ： Whole-system-level behavior-based analysis is more accurate than tranditional signature-baed for malware detection. Behavior-based analysis can be used to expose the system effects after executing malware. By providing service trustworthiness and cloud security, our analysis platform can examine cloud storage automatically to secure the cloud application. To this end, we integrate the VM-based malware analysis with cloud computing for both accuracy and efficiency. This project developed a proactive service for the Internet to collect and analyze malware. The analysis results can provide a security judge for users before file download, and the whole-system level behavior-based analysis is helpful verify developed software for security testing. There are two major accomplishments of this research project: (1) To extract the behaviors of malware by

using whole-system-based analysis, (2) To design and develop a cloud analysis platform which benefits on the ability of cloud computing. The cloud analysis platform can automatically collect and analyze executables on the Internet, and save the analysis result as reports which are stored in distributed database. Due to the scalability of computation on the cloud, the platform can provide better performance for analysis. In this project, we developed a security analysis platform and a whole-system-wide analysis tool. Both of them are aim to extract the malicious behaviors of malware for further malware research.

英文關鍵詞： Cloud computing, Malware behavior analysis, Malware detection

行政院國家科學委員會補助專題研究計畫 ■成果報告 □期中進度報告

前瞻性雲端安全儲存、防護、行為分析與觀測平台--子計畫二：基於機器碼之 Windows 惡意程式行為分析雲端平台(2/2)

計畫類別：■個別型計畫　□整合型計畫
計畫編號：NSC 100-2218-E-009-004－
執行期間： 100 年 8 月 01 日 至 101 年 7 月 31 日

執行機構及系所：國立交通大學資訊工程學系（所）

計畫主持人：謝續平
共同主持人：

成果報告類型(依經費核定清單規定繳交)：□精簡報告　■完整報告

本計畫除繳交成果報告外，另須繳交以下出國心得報告：
□赴國外出差或研習心得報告
□赴大陸地區出差或研習心得報告
□出席國際學術會議心得報告
□國際合作研究計畫國外研究報告

處理方式：除列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

中　華　民　國　101 年　10 月　31 日

摘 要

全系統層次的惡意程式行為分析技術可以比以往傳統靜態分析技術更精確地判斷一個執行檔案行為是否影響系統安全。本計畫擬提供一個可信賴性的分析服務與雲端應用層的安全防禦做結合，藉此為雲端服務技術做安全性的把關。本研究將探討此技術如何結合虛擬機器技術與雲端運算，以適用於分析工作的分派，使檔案可以自動地、大量地透過連續的分析檢驗來達到程式行為分析。有別於傳統的軟體即服務（SaaS）模式，本計畫除了提供可疑檔案上傳平台介面外，還開發出一個較為主動的檢驗服務，針對網際網路上現有檔案進行蒐集與分析，除了可供使用者判斷下載檔案的安全性，也可讓資安人員用來驗證其開發出的檢測軟體是否做出正確的判斷。

本研究計畫共有兩大研究成果：（一）使用全系統層次的行為分析技術來擷取惡意程式行為以達到更準確的分析、（二）開發與設計雲端架構分析平台以提高安全分析產能。雲端架構分析平台可以自動地蒐集網路上的惡意程式樣本進行分析，並且對其行為的分析結果利用分散式資料庫儲存。而分析的階段皆以自動化、平行化的方式來增加運算節點的使用率以提高分析的產能。本計畫將會實際設計與開發一個雲端架構分析平台，並會對惡意樣本做行為分析，以利用其惡意行為來偵測未知的惡意程式樣本。

關鍵詞

惡意程式、程式行為分析、雲端安全

Abstract

Whole-system-level behavior-based analysis is more accurate than tranditional signature-baed for malware detection. Behavior-based analysis can be used to expose the system effects after executing malware. By providing service trustworthiness and cloud security, our analysis platform can examine cloud storage automatically to secure the cloud application. To this end, we integrate the VM-based malware analysis with cloud computing for both accuracy and efficiency. This project developed a proactive service for the Internet to collect and analyze malware. The analysis results can provide a security judge for users before file download, and the whole-system level behavior-based analysis is helpful verify developed software for security testing.

There are two major accomplishments of this research project: (1) To extract the behaviors of malware by using whole-system-based analysis, (2) To design and develop a cloud analysis platform which benefits on the ability of cloud computing. The cloud analysis platform can automatically collect and analyze executables on the Internet, and save the analysis result as reports which are stored in distributed database. Due to the scalability of computation on the cloud, the platform can provide better performance for analysis. In this project, we developed a security analysis platform and a whole-system-wide analysis tool. Both of them are aim to extract the malicious behaviors of malware for further malware research.

Keywords

Cloud computing, Malware behavior analysis, Malware detection

目 錄

壹、計畫內容及目的

一、前言及背景說明

由於雲端計算能有效利用實體機器的運算資源，藉著聯集所有相連接的運算節點來增加安全分析的產量。防毒軟體公司早在進行郵件過濾時便發現，以傳統的方式如透過特徵偵測或靜態分析技術先取得惡意程式之特徵，再將此特徵用於系統中的檔案或是網路上的封包做掃瞄比對，可以十分準確且快速的達到阻擋攻擊的效果。因為目前病毒的數量更為龐大、變化又快，病毒碼擷取特徵的方式來處理惡意程式已成效不彰，所以借助雲端運算的能力去收集大量的樣本並進行資料探勘以找出有用資訊，以解決病毒防疫空窗期與防毒軟體日益肥大的問題。

除此之外，新型的攻擊手法使傳統病毒碼的偵測效益降低。先進的惡意程式撰寫技術，加殼（Packing）、多型（Polymorphism）、變形（Metamorphism）、核心等級匿蹤技術（Kernel Level Rootkit）等，會讓同一種行為的惡意程式有不同的病毒特徵。故國內外學者在近年來皆朝著動態分析方式來萃取惡意程式的行為特徵，而不再是靜態的二進位執行檔案之特徵值。為了更全面的分析，越來越多學者開始採用多面向的分析方式來做惡意程式偵測。目前分析技術可分成兩大類：（一）靜態分析（static analysis）、（二）動態分析（dynamic analysis）。靜態分析將能夠針對檔案的內容做檢測，希望能夠找出不應該存在於檔案內容，或是能夠判斷該檔案是否有可疑的加殼加密行為。此部分主要以特徵比對（signature-based）來做判斷。此外，動態分析而是直接模擬運行時的狀態，若該檔案為可執行檔時，我們將利用已知的動態堆疊資訊取得技術，來檢測該檔案是否在執行時期，有取得堆疊資訊的能力。由於動態分析與靜態分析能夠互相彌補之不足的地方，所以本計畫目標將設計一個分析平台，藉此平台能夠結合多種分析技術，已達到較完整的檔案檢測分析。

本系統架設在一個不對外公開的區域網路之內，屬於一種「內部雲 (private cloud)」的雲端應用。不同於國內大部分雲端系統，業者或是系統管理員只能夠提供Hadoop運算的帳號，其「公開雲 (public cloud)」的系統架構並不適用於惡意樣本分析平台，原因有下列敘述：第一，動態程式分析不能夠利用MapReduce來平行化其動態分析的流程，因為程式被執行的控制流程是相互關聯的，在不同的系統狀態，即使給予相同的指令也會有不同的結果產生，所以不能把動態程式分析分散處理。第二，提供虛擬環境來讓租用者分析惡意程式，可能會造成誤判。近年來已經有出現偵測虛擬環境的惡意樣本，如果在分析環境都只有在虛擬環境之中，則該樣本將永遠不會執行其惡意程式碼。最後，公開雲端架構往往需要複雜的使用者權限管理，而因為資料分享共用的原因，個人與系統的安全性也會面臨考驗。如果此系統架構太過於公開，很有可能被有心人士入侵，修改其分析結果。綜合以上三點，本系統要提出一個利用雲端運算所帶來的高產量、可擴充性和高儲存能力，來實作一個樣本分析平台。

雲端分析平台並不需要有繁雜的使用者管理，它可以被視為一個惡意程式分析的叢集式電腦。主要的架構包含分散式資料庫、檔案輸入介面和惡意樣本分析程式。而蒐集到的樣本資料可以利用Hadoop做資料統計或是更深入的特徵碼分析。分散式的資料庫目前以Facebook團隊開發的Cassandra作為基礎，其讀寫速度與資料庫的大小只有常數關係，並不會因為資料庫太大而造成寫入速度變慢。而檔案輸入介面，可包含網路爬蟲以及使用者上傳兩大部分。網路爬蟲則

是利用Nutch (利用Hadoop實做的搜尋引擎建立系統) 來蒐集散播在網路上的資料，當作本系統的輸入。除此之外，還可以接收使用者上傳的檔案，提供雲端檔案安全性檢驗的服務。惡意樣本分析程式將會整合先前所開發的動態程式分析工具，其分析項目有包含靜態可執行檔分析、檔案文件安全性分析和利用虛擬機器觀測程式行為。惡意程式樣本分析可以持續的開發，有彈性的掛載在本子系統上，以檢驗雲端的檔案或是使用者欲檢測的檔案之安全性。

二、 文獻探討與相關研究

惡意樣本的蒐集已是資安相關領域中相當重要的問題，許多樣本蒐集平台也紛紛被開發出來[1][2][4][5]。以往的樣本蒐集大多採取 Honeypot 的方式來誘捕攻擊者，分析其一連串的入侵手法以及殘留在電腦系統中的惡意檔案。但利用 Honeypot 蒐集的效率並不高，故有學者紛紛提出增加被攻擊的機率以提高樣本蒐集的速率[3]。由於雲端技術的誕生，使用者可以靠著自行架設的機器來擷取網路的檔案，並且可以多台同時合作和達到橫向擴展(Horizontal expandable)來達到運算的可擴充性。利用 MapReduce 的雲端計算架構來平行化擷取網路檔案，可以讓使用者在最短的時間內建置自己檔案樣本來源。

近年來，惡意程式樣本分析也因虛擬機器的發展，而有了更多分析的可能性。透過虛擬機器所提供的隔離性與對系統的完全控制，可以讓分析人員深入的了解其資料流向以及對系統的影響。這些利用虛擬機器分析的系統[10][11][12][14]，因為需要龐大的運算能力來模擬整個系統的運行，所以無法大量的被應用在現實生活中。尤其是在樣本儲存的架構中，傳統的 SQL-like 的資料庫已被證實其存取速度會因項目數量增加而減少。在分散式儲存系統中的 Cassandra 利用 DHT(Distributed hash table)的方式來儲存資料，除了存取時間不與資料量的大小成正比，還可以輕易的擴充儲存節點，非常適合用來儲存惡意樣本。儲存在資料庫中的惡意樣本除了可以供研究使用，也可以利用分群的技術來實作未知樣本的惡意行為偵測[6][7][8][9]。

結合分析以及樣本蒐集的惡意樣本分析平台，並非本系統的獨特創新[13][15][18]。這也驗證了此系統的可行性之高，並且可以實際的運用在病毒分析與偵測的領域。本系統的創新在於如何簡易的掛載不同種類的分析工具，甚至往後可以推廣到其他相關研究或是提供樣本給學術界研究分析。為了在為期一年的開發期完成，我們評估了不同種的工具系統，並且修改和整合其原始碼至此子系統。除了整合現有的工具，還實作一些機器管理機制以及節點間通訊協定，讓此系統更加有彈性。在實作部分，我們利用高階語言來物件化功能、提高可讀性和簡易化偵錯以縮短開發時程。

在惡意程式分析的方法中，本系統將模擬特殊的記憶體與暫存器架構，在記憶體與暫存器額外附上一個 bit，用以標記該暫存器或是記憶體所存的值是否遭受污染(tainted)，1 為 tainted，0 為 clean。在此系統會將不是由程式本身產生的記憶體位址覆寫標記為 taint，並監控程式的執行過程中，是否由已標記 taint 的記憶體或暫存器感染其它更多的記憶體和暫存器，並觀察系統是否執行已標記 taint 記憶體區塊內容。

觀察現今多數惡意軟體，常常不以癱瘓對方電腦為主要目的，而是以竊取對方隱私資訊為首要目標，如：間諜軟體與側錄程式…等。這些惡意程式不經使用者同意，將竊取來的私密資訊透過網路封包傳送給惡意攻擊者。因此，在 Panorama[25]的方法中提出，若將使用者的隱私資料標示起來，並監控存取此資料的程序或應用軟體，記錄其存取途徑是否有任何的惡意行為，

如：不經使用者許可將資訊經過網路封包傳遞，若發現這一類程序，則該程序極有可能是惡意程式。透過這種方式，便能以自動化的方式檢測是否有可疑的間諜程式夾雜於應用程式當中。

此外，惡意軟體常運用 Hooking 技術來控制惡意程式的執行流程，讓惡意程式等待使用者執行某些特定的動作後，(如：當使用者敲擊鍵盤時，鍵盤側錄軟體便開始計記錄使用者所敲擊的資訊，或者當惡意程式開始執行時，會通知 rootkit 將自己的程序隱藏起來)。因此如果能在記憶體中找到這些惡意程式的掛載點，便很有機會檢測到惡意程式的存在。現今大多數檢測軟體的作法是觀察程式執行記憶體中是否存在惡意軟體中，通常會包含的 Hooking 特徵，這類軟體包括 VICE[36]、System Virginity Verifier[37]。假若檢測結果發現應用程式中包含特定 Hooking 特徵碼，則代表該程式有可能是惡意程式，但如果惡意程式使用一個未曾見過的 Hooking，則無法被偵測出來。因此 HookFinder[38]提出的方法中將模擬一個系統，嘗試標記惡意軟體所帶來的更動，如：機碼的更改。這些更動很可能是由某個 Hook 所為，接下來系統會監控整個程式的執行流程。如果發現 X86 CPU 的 Program Counter 所載進來的記憶體位置，是因為剛才的更動所造成的，則代表目前的程式可能正嘗試要執行惡意軟體，因此可以判斷目前執行的程式極有可能夾帶惡意程式。

惡意軟體為了隱藏自己的存在，不被防毒軟體偵測，因此常透過加殼的方式來打亂程式特徵碼，逃過反向程式追蹤。只有在程式執行的過程中，才依序將程式解譯出來並寫入記憶體執行，因此在執行前，無法透過特徵碼檢驗的方式發現此類惡意程式。目前已有許多研究希望能夠以自動化的方式解譯惡意程序正確執行順序的方法。不過目前為止，除了特定已知的解譯方式，還無法針對特殊加殼的方式進行解譯。由於執行任何程式之前，必須先將該程式載到記憶體中才有可能執行，惡意程式當然也不例外。因此在 Renovo[39]的方法中提出，使用模擬環境，讓目標程式在模擬系統上執行，並監控所有進行記憶體寫入與程式執行流程變更的指令。此外，開闢一塊虛擬的記憶體區塊，標記程式執行過程新寫入記憶體的程式碼，最後將這些程式碼依序 Dump 出來，以取回程式碼執行的正確順序，便能對程式碼進行特徵碼比對。

動態汙染分析除了使用軟體實作外也可以使用硬體實作，例如：MIT 的 Computer Science and Artificial Intelligence Laboratory 提出建構一個特製處理器去實作的方法[41]，在處理器中的暫存器都有一個額外的位元去記錄該暫存器的值是否有被 taint，而記憶體中可被存取的最小單位也有一個功能相同的額外位元。在[42]中也提出一種建構在 Raksha[43]系統上的實作方法。資料一開始進入系統時就會根據設定的規則去判斷要不要 taint 資料，若是 taint 的資料被拿來當作指令執行或是跳躍指令的目標位址，則可判定為惡意攻擊。由於使用硬體來實作，速度快但是成本花費太高；若使用軟體來實作，則雖可降低成本但是效率不佳，所以我們選擇使用軟體實作再加上平行處理的技術，這樣就可以提高效率及降低成本的目標了。

三、 計畫目的

本研究執行之目的有二:(一)使用全系統層次的行為分析技術來擷取惡意程式行為,(二)開發與設計雲端架構分析平台以提高安全分析產能。以下對此二大目的加以說明：

■ **使用全系統層次的行為分析技術來擷取惡意程式行為**

由於雲端計算在應用上提供平台服務（PaaS)，不管是客戶或是終端使用者皆有可能放置檔案到此平台，並藉由此平台進行傳遞、交換、散播等動作，這些方式與傳統網際網路上檔案交換並無太大不同，造成雲端平台上也有傳統網路攻擊的疑慮。對於雲端提供商與雲端平台客戶來說，若是在雲端上發生類似傳統網路的蠕蟲攻擊，營運上的損失與對商譽的傷害將是無法想像的巨大。因此，放置在雲端平台上的檔案安全與否變得格外重要。而如背景所述，傳統靜態分析對於雲端平台上千變萬化的檔案已經不夠及時，必須等待特徵碼產生後才能提供為判斷的依據。

因此本研究計畫之首要目標為，開發與整合多項檔案安全分析技術，並將其融合至雲端分析平台以顧及多面向的雲端檔案。首先，分析雲端平台上的目標檔案是否內藏加殼程式來躲避分析。若目標檔案為可執行檔，則進一步利用全系統層次的行為分析技術來詳細分析該執行檔是否可能對使用者系統造成危害，並進一步產生報告。有了以上的安全分析，可保護雲端服務平台，降低使用者上傳的未知檔案傷害風險。

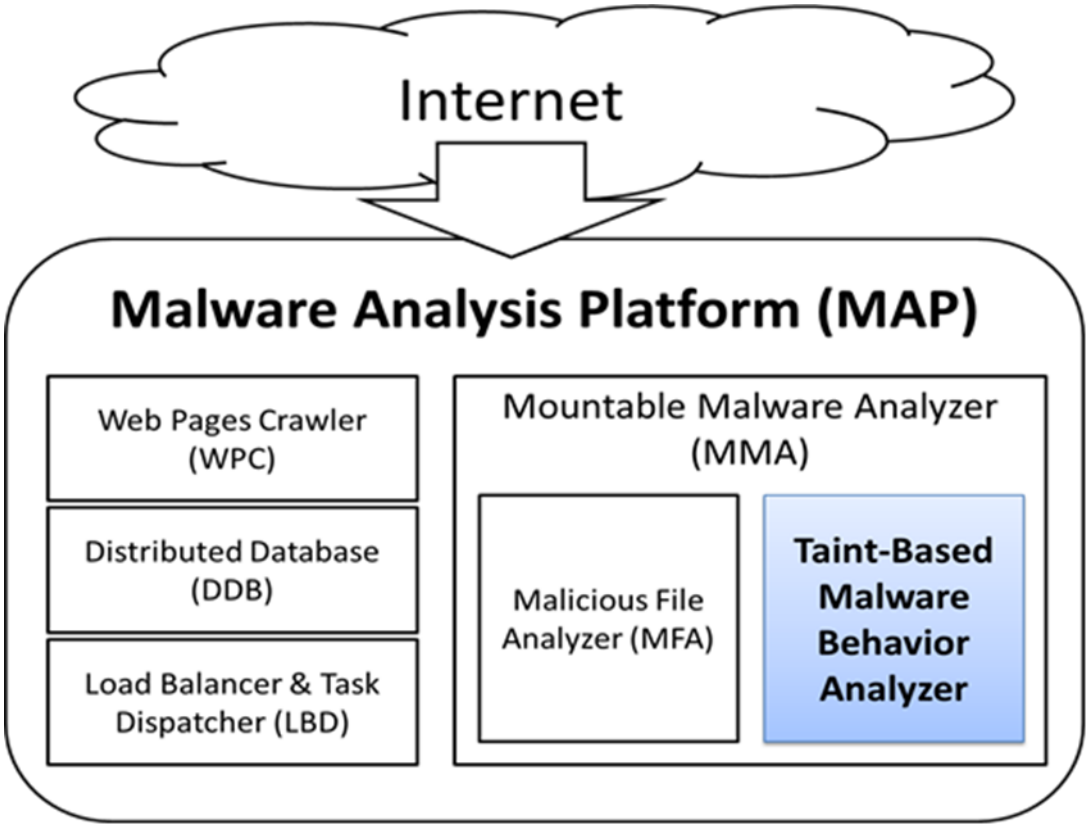
■ **開發與設計雲端架構分析平台以提高安全分析產能**

檔案分析的方式眾多，越來越多不同面向的分析工具大量的產出。這些不同面向的分析工具因為分析的方法不同而有互補的作用。靜態分析與動態分析各有其優缺點，靜態分析雖然運算負擔較動態分析小，但卻無法得知執行時期的資訊，所以在分析應用上較適合特徵式偵測。另一方面，動態分析除了負擔較大的運算開銷以外，分析範圍需要大量的運算時間搭配，故並沒有一種方法是完美的。如果能夠簡易的整合市面上各種分析工具而達到互補作用，又可有效且平均的分配至多台電腦，則可以加強分析能力又可縮短分析時間。

為了增加分析的效率與產出，需開發一個雲端架構式的分析平台，把目前現有的分析工具模組化，依使用者需求掛載分析模組且動態的去管理。這個平台必須具備有：一）可擴充性、二)自動化、三)高容錯性。可擴充性可以讓本分析平台藉著增加運算節點來達到效能的提升，有效的利用運算資源。然而，管理者需要同時管理多台電腦，而此系統的眾多繁雜的設定檔，將會是往後維護人員的最大問題，自動化的管理與設定將會使系統更方便使用。由於在雲端架構中有需多節點會交互溝通，容錯性可以讓系統工作得更久更穩定。我們將實作與設計一套系統架構，符合雲端架構之精神來實作該分析平台。除了可以提供分析服務以外，該平台還有自行蒐集網路文件檔案的能力，可以讓資安人員測試自行開發的分析工具，提早搜集網路上行為可疑的文件檔案。開發人員可以不用費心去管理其分析模組與複雜的架設系統，也可以輕鬆快速地擁有客製化的分析結果。

四、 研究方法

本分析平台主要由四個部分所組成：一）檔案來源子系統、二）儲存子系統、三）分析器管理子系統、四）全系統層次的行為分析系統，彼此間的關聯性可由**錯誤! 找不到參照來源。**及**錯誤! 找不到參照來源。**的架構看出。本平台主要分成兩大流程(圖表 10)，第一，透過檔案來源子系統來取得分析所需的檔案資料，接著依照固定的格式放入儲存子系統中。第二，分析器管理子系統從儲存子系統中取得待分析的檔案資料後，開始進行分析。全系統層次的行為分析系統為分析器管理子系統中的其中一模組，亦為本計畫的重點開發項目。以下將分部介紹之上述四大子系統，並在最後以一個正規化的模式表現之。



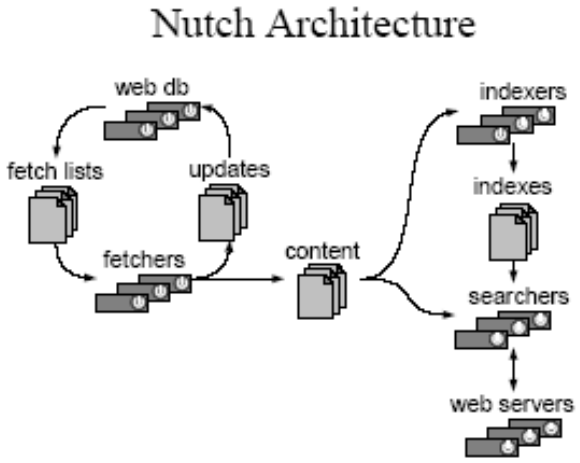
圖表 1 本系統架構圖

- **檔案來源子系統**

本系統需要接收大量的檔案作為分析的對象，而此系統必須具備以下特色：（1）處理大量檔案的能力、（２）各種不同的檔案來源，例如：網際網路、檔案系統、P2P 網路、E-mail…等，以分析並取得各種不同型態的惡意軟體、（３）適用於各種不同的使用方式，例如：網路爬取、使用者上傳。為滿足以上需求，本系統實做一個 PushTask 之函式庫，此 API 可將任意檔案輸入本系統進行分析，並指定該檔案的優先權。此 PushTask 函式庫先產生檔案的工作識別（UID），在將此 UID 放入 Redis 中，並將檔案內容存入 Cassandra 中，將檔案輸入我們的系統中。利用此 API 為基礎，我們實做了兩種檔案來源系統：網頁爬取系統和批次檔案輸入。

在網頁爬取部分，我們使用了 Nutch 為爬取的開發工具基礎。Nutch 為 Apache Fundation 名下的開放原始碼計畫，是一套由 Java 建構的網路搜尋引擎軟體，具有相

當良好的跨平台特性。並且支援 hadoop 系統，具有良好的可擴充性。Nutch 也提供了許多 API 可以使用，分便於系統的架構開發。



圖表 2 NUTCH 系統架構

圖表 2 為 Nutch 的系統架構，如圖中所示 Nutch 可以分為網頁爬取（Crawler）以及搜尋引擎（Searcher）兩部分。網頁爬取部分由一組使用者定義好的初始網頁（seed）做為搜尋的起點，搜尋與起始網頁相關聯的網頁，並將網頁的連結資料取出（update）建構成新的初始網頁（web db）進行下一輪的搜索。在本系統中，只需要對網頁做抓取（圖表 2 左半部）的動作，而不需要進行索引以及搜尋的部分，因此我們利用 Nutch 提供的 API 進行網頁爬取並省略做索引搜尋的部分以增加系統效能。並透過 hadoop 系統，藉由 mapreduce 將運算分散至各節點進行運算，增加整體產出。

在 Nutch 爬取大量網頁時，會抓取許多不同格式的檔案，如：exe、jpg、zip、html…等，而我們系統目前主要是以分析可執行檔為主，因此在爬取網頁檔案時，僅保留系統可以分析的檔案格式（exe、zip、rar），過濾掉其他無法分析的檔案格式。在 Nutch 抓取檔案後，會執行 PushTask 函式，將檔案輸入到我們的系統中做分析儲存。Crawler 部分為此系統一長期穩定的檔案來源，僅具有最低程度的優先權。

在批次檔案輸入部分，我們撰寫一支程式將一個資料夾下的所有檔案透過分析平台的函式庫輸入到分析平台上做分析。此程式會搜尋所有資料夾下的檔案，並產生該檔案之工作識別（UID），並將工作識別放入 Redis，在將工作識別連同檔案內容放入 Cassandra 做保存。檔案系統輸入這部分是將檔案系統上的資料輸入到我們的系統做分析，例如：單一資料夾、或遠端掛載的資料…等，這部分主要是應用在批次處理資料，會需要一定的反應時間，應此在系統中具有高於 Crawler 的優先權。

- **儲存子系統**

在儲存系統方面，由於此研究已經被探討多年，市面上也有大量的資料庫可以應用於本分析平台上。為了節省開發時間以及維護方便，經過深入地了解以及搜索之後，我們選擇了兩個現有的資料庫系統 Cassandra 和 Redis，透過修改程式、設定檔案和內部些許的程式碼，快速地達到本分析平台所需要的儲存能力。

Cassandra 是一套分散式的資料庫系統，在分析平台中用來儲存未分析的檔案內容以及分析完的結果。會選用 Cassandra 的考量在於以下三點。

1. **NoSQL**：Cassandra 是採用 Key-Value 的儲存方式，因此在資料庫有大量資料的情況下，讀寫速度遠快於一般 SQL 的資料庫。此外，Cassandra 在資料庫的欄位也有別於 SQL 資料庫，通常 SQL 資料庫的欄位必須在一開始就訂定，之後如果有新增或移除，則必須重新處理資料庫的內容，而 Cassandra 正好相反，它能動態的新增或移除欄位，讓整個分析平台在新增分析模組的情況下變得更為簡單。
2. **系統穩定度高**：Cassandra 的分散式是採取 ring 的架構,因此每台 Cassandra 都是相同的身分，意即非主從式(Master-Slave)的架構，所以當任何一台資料庫毀壞，也不會造成整個儲存系統停擺。另外，在新增資料庫機器時也不需要將資料庫全部關閉，所以能使整個分析平台的執行更加穩定，不需要因為新增機器而暫停運作。
3. **可與 Hadoop 整合**：Cassandra 在 0.7 版以後提供與 Hadoop 整合的功能，能將資料庫的內容直接進行 Map-Reduce ，因此分析平台系統在後期的發展能將分析完的結果做進一步的處理。

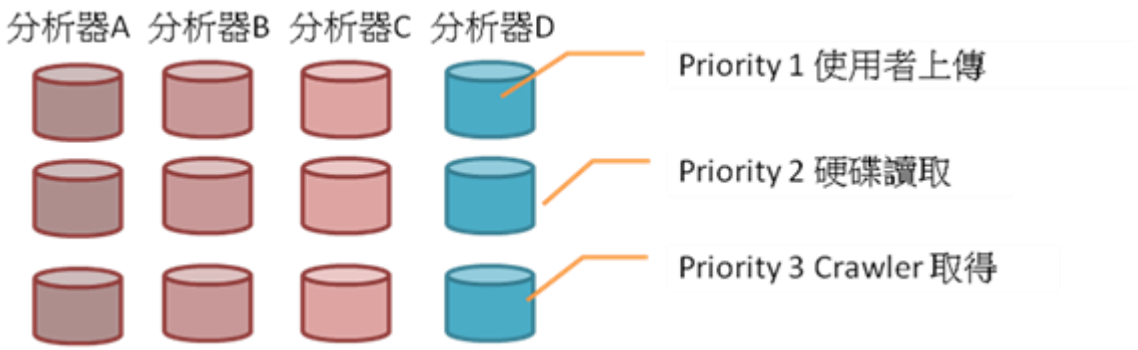
由於以上考量，本系統將 Cassandra 當作是整個系統的後端，專門儲存所有的檔案以及其結果，相較於稍後會提及的 Redis，Cassandra 所存放的資料是屬於永久性的。

Redis 是一單機的資料庫系統。在分析平台中以佇列(Queue)的模式來儲存未分析的檔案的工作識別號碼，而分析模組則從此處拿取工作識別號碼來向 Cassandra 取得未處理的檔案。選用的考量如下：

1. **讀寫快速**：Redis 將所有的資料庫內容存放在記憶體中，並且定期會備份至硬碟內。一般而言，硬碟的讀寫速度小於記憶體的讀寫速度，所以在考慮儲存暫時性的過渡資料（例如:工作識別號碼）時採用 Redis 而不是 Cassandra。
2. **豐富的資料結構支援性**：Redis 本身支援 List（有序） 和 Set（無序）兩種資料結構，同時也有實作排序的功能，因此在本系統的開發過程中就運用到其 List 功能以及不同的檔案來源來實作出工作優先權佇列（Priority Queue），將其運用在工作識別號碼的分配。下面詳述。

本系統在 Redis 部份實作了工作優先權佇列，架構如圖表 3，其主要目的是為了因應不同的檔案來源而給予不同執行順序的安排。分析器 A 到 D 代表不同種類的分析器，每種分析器都有三個佇列，佇列則是用 Redis 的 List 資料結構實作而成，每個佇列分別代表不同的檔案來源:

使用者上傳、硬碟讀取和 Crawler 取得。當分析器拿取工作識別號碼時就由優先序高（越小越高）的拿起。



圖表 3 工作優先權佇列於 REDIS 內的實作概況

■ **分析器管理子系統**

在本系統中，可以掛載多個分析器。藉由多個分析器的分析結果，以取得更準確的結果。因此我們需要建構一套分析器管理子系統來管理各種不同的分析器，讓各種不同輸入輸出的分析器能正常運作於本系統中。分析器管理子系統具有以下功能：

1. **自動化管理分析輸入輸出：**透過將輸入輸出流程自動化，可以降低分析器開發者測試所需時間，也可為此系統取得大量分析結果可供使用比較。本子系統會自動從資料庫抓取分析檔案，並在執行完成後自動將結果存回資料庫，可以達成自動化管理分析輸入輸出。
2. **快速方便的安裝、起動、停止分析器，易於管理：**藉由快速安裝分析器，可以將分析器快速的安裝在新節點之上，增進系統可擴充性。而快速啟動停止分析器，可以增加本系統運作上的彈性，進而達到分析器的動態調整。本子系統會自動讀取分析器開發者所撰寫的設定檔，根據該設定檔可以讓我們快速的安裝起動分析器。而在起動時，系統會記錄分析器執行時的 PID，系統便可依此停止分析器。
3. **整合各種不同形式的分析器：**由於惡意軟體的種類眾多，並無任何可以偵測所有惡意軟體之技術。因此需要使用多種不同形式之分析技術，以取得最完整的結果報告及準確的偵測率。本子系統透過一個統一的設定檔，來記錄不同形式分析器的使用方式，讓系統可以整合各種不同形式的分析器。
4. **執行多個分析器於單一節點，以增進系統產量：**由於各種分析器執行時所消耗的系統資源不一，而單一一台分析器也鮮少用盡系統資源。因此透過同時執行多個不同分析器於單一節點上，增加系統資源的使用率，並促進系統產量。在此子系統中，我們可以動態增減執行的分析器，並同時分派任務執行，以達到單一節點執行多分析器。

欄位名稱	欄位說明	範例
COMMAND	執行分析器時所需的指令	COMMAND= ./scanner.sh
LOG	分析報告的儲存位置	LOG = clamav.out
TYPE	分析器的名稱	TYPE=ClamAV

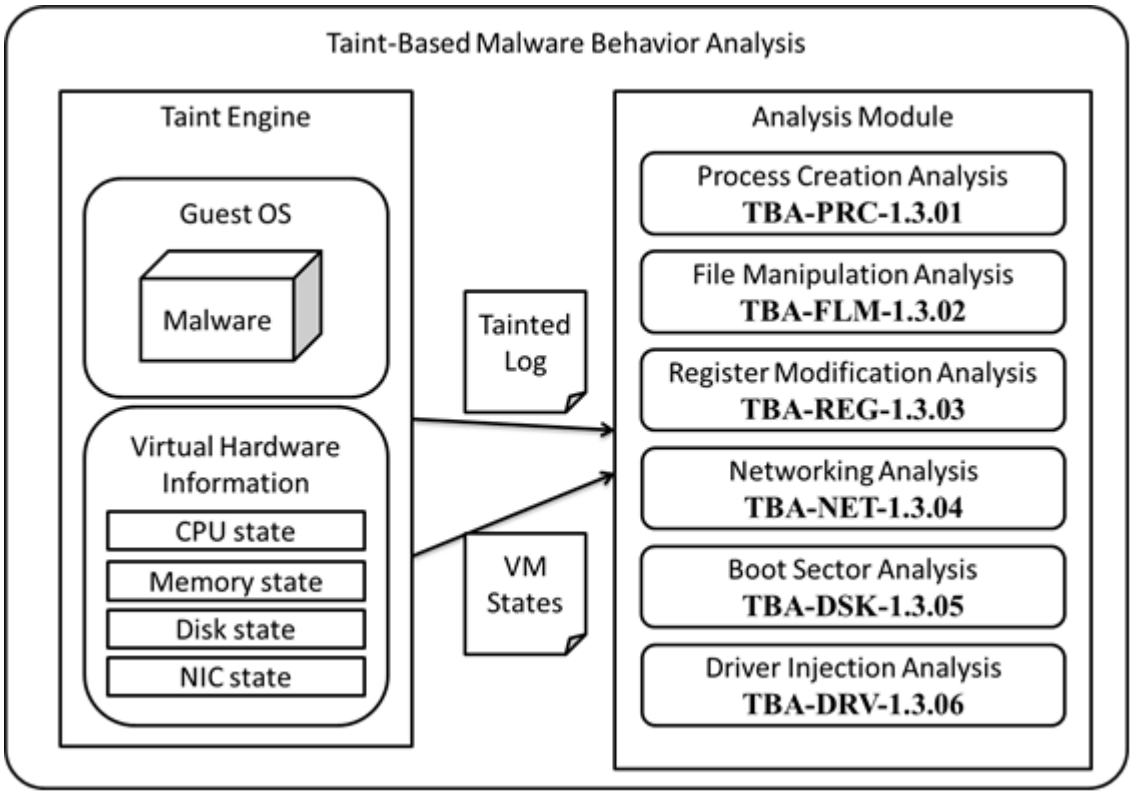
圖表 4 分析器設定檔內容

要將分析器運行於本系統中，只需要編寫一個額外的設定檔（config），以 key-value 的形式將分析器的啟動以及結果位置做設定，如圖表 4。 透過設定設定檔中的 command 欄位（如：COMMAND= ./scanner.sh），可以指定分析器執行的方式，本系統便可自動分配檔案並且執行分析。log 欄位則用來指定最後的結果存放位置（如：LOG = clamav.out），本系統便會自動保存分析結果。

我們實做 AnalyzerInvoker.rb，搜尋分析器目錄裡的設定檔（config），取得各分析器的啟動方式以及輸出檔案。接著透過先從 Redis 中的佇列取得要分析的工作識別後，以該工作識別去查詢 Cassandra 資料庫中抓取相對應的檔案。並利用設定檔中的起動方式將分析器啟動。分析完畢後，則依據設定檔中的分析結果位置取得分析報告，將最後結果回存到資料庫中。

- **全系統層次的行為分析系統**

在惡意程式分析系統部分，本計畫已開發一分析系統，藉由著整合虛擬機器與程式行為分析來達成揭露隱藏行為的惡意程式。揭露方式主要以登錄檔、檔案系統、網路行為等為主，利用去年開發的分析平台繼續增強其功能性。



圖表 5 全系統層次的行為分析系統

偵測程序修改行為：本子系統為了達到偵測所有程序的行為，進一步分析惡意程式的攻擊行為，此實做系統將針對兩個方向進行分析：是否有新增的程序、是否有隱藏程序存在。基於虛擬機器的應用上，根據虛擬機器內所得到的程序資訊以及從虛擬機器外取得系統資料並分析後的資訊，進一步比對兩方程序紀錄是否存在差異，藉此來偵測出是否有可疑未知程序產生以及隱藏程序的存在。我們將在 LINUX 環境下以 QEMU 建立虛擬機器，當欲檢測之檔案輸入時，將其運行在虛擬機器上，再利用 QEMU 的 SNAPSHOT 功能來協助實做此系統。

偵測檔案修改行為：運行惡意樣本(ROOTKIT)之後可能讓系統查詢檔案的函式遭受竄改。故無法在系統內部直接做偵測。為了達到精準的偵測，我們透過虛擬磁碟上的內容，逐一的將檔案萃取出來比對。利用修改前後的差異性，來得知那些檔案被修改、創建、刪除。本子系統利用此資訊來達到偵測檔案修改行為之目的。為了找出系統內是否有被 ROOTKIT 隱藏的惡意檔案，本系統運用了 TSK (THE SLEUTH KIT)來協助我們偵測該行為。此套件可以觀察虛擬硬像檔的檔案系統裡所有資訊，包含有目前硬碟裡的附加資料（METADATA）、或檔案的內容等。找出被隱藏的檔案，GUEST OS 執行可疑程式時，遇到要修改檔案時，在 HOST OS 透過 QEMU 把要修該檔案的磁區號碼記錄下來，利用 TSK LIBRARY 查詢硬碟映像檔的 FILE ALLOCATION TABLE 找出存在該磁區的檔案名稱及檔案路徑，再去 GUEST OS 利用 WIN 32 API 查詢是否有該檔案的存在，若沒有，則表示該檔案被隱藏。為了找出開機磁區是否有被修改，GUEST OS 執行可疑程式時，遇到要修改檔案時，透過 QEMU 把要修該檔案的磁區號碼記錄下來，如果修改的是第一個磁區(開機磁區)，我們則判斷有開機磁區修改的行為(預計整體流程如下圖）。

偵測登錄機碼修改行為：子系統將會精準地列出所有被更動過之登錄機碼。因為惡意程式往往會修改系統上的登錄檔，在開機時啟動惡意程式。除此之外，ROOTKIT 還會竄改 WIN32 函式庫或系統呼叫（SYSTEM CALL），以避免某些新增之機碼被發現。本系統將直接讀取硬碟中用以儲存這些登錄檔的 HIVE 檔案，並透過自己的程式加以解析。惡意程式為了讓自己在系統重開機後仍然能夠運作，必須修改登錄機碼並指定開機啟動程式的目錄，甚至為了比防護軟體獲得更早啟動的優先權，會把自己註冊成驅動程式或系統服務。要達成這些目的，系統登錄檔往往是惡意程式必須更動之處。然而，防護軟體或使用者自身常常會對這些地方作基本的檢查，如 HKEY_LOCAL_MACHINE\SYSTEM\CURRENTCONTROLSET\SERVICES 或 HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\POLICIES\EXPLORER\RUN 這些機碼，都是現今最常見的檢驗點。因此，現今的惡意程式程式為了避免被發現，就必須要使用 ROOTKIT 技術來隱藏這些特定的機碼。在 WINDOWS 程式中，多數的登錄檔編輯軟體在對這些登錄檔進行查詢時，是使用 WIN32 程式庫所提供的 API 如 REGENUMKEY, REGENUMVALUE 或是 WINDOWS 本身的 SYSTEM SERVICE 如 DIRECTNTENUMERATEKEY 或 DIRECTNTOPENKEY 等等，這些函式往往被已經侵入系統的 ROOTKIT 進行 HOOKING 的動作，因此在查尋這些登錄資料時，將完全無法發現某些被隱藏的項目。我們分析的步驟如下，首先我們將儲存登錄資訊的幾個重要 HIVE 檔案，從虛擬機器的硬碟映像檔中抽取出來，這些檔案包括：

C:\WINDOWS\SYSTEM32\CONFIG\SAM
C:\WINDOWS\SYSTEM32\CONFIG\SECURITY
C:\WINDOWS\SYSTEM32\CONFIG\SOFTWARE
C:\WINDOWS\SYSTEM32\CONFIG\SYSTEM
C:\WINDOWS\SYSTEM32\CONFIG\DEFAULT
%USERPROFILE%\NTUSER.DAT

請注意以上的抽取動作皆由系統外部進行，而不經由系統內部的 API 或系統呼叫。當以上檔案從硬碟映像檔中抽取出來後，我們將解析這些 HIVE 檔案，並列出某些重要登錄目錄下所有的機碼值，如：

HKLM\SYSTEM\CURRENTCONTROLSET\SERVICES
HKLM\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\POLICIES\EXPLORER\RUN

HKLM\SYSTEM\CURRENTCONTROLSET\CONTROL\SESSION MANAGER\BOOTEXECUTE
HKLM\SOFTWARE\MICROSOFT\WINDOWSNT\CURRENTVERSION\WINLOGON\USERINIT
HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON\SHELL
HKLM \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUNONCE
HKLM \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN
HKLM \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUNONCEEX
HKLM \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUNEX
HKCU\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUNONCE
HKCU \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN
HKCU \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUNONCEEX
HKCU \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUNEX

等等，之後由系統內部的觀察程式，以 WIN32 API 去取得同樣目錄下所有的機碼值，並與系統外部的觀察結果進行比對，以確認是否有隱藏機碼的 ROOTKIT 行為。

偵測網路通訊行為：為找出程式是否會開啟網路通訊阜並隱藏其行為，需要比對不同時間先後的通訊阜列表。列出系統中所有被打開的通訊阜，透過篩選出最近新增的，這些新增的通訊阜即是該程序所開啟的。而這些列表往往常駐在記憶體中，且由系統保護著，並無法用一般 USER-MODE 存取。所以我們利用虛擬機器可以隨意的讀取 GUEST 端記憶體的特性，把記憶體的狀態全都記錄到 HOST 端的硬碟。參考現有的 ROOTKIT 技術，在記憶體中找出系統資訊，來得到正被載入執行的通訊阜列表。現有的 WIN32 系統架構裡，其存放的資料結構固定，使其可以利用 SIGNATURE 的方式來偵測記憶體裡，所存放的通訊阜開啟列表資訊。新增的通訊阜開啟列表，會存在於兩個不同的時間點的清單差集內。整體流程大致如下：

- 把欲偵測的程式放到虛擬機器裡。
- 把虛擬機器的記憶體全部記錄下來(MEMORY DUMP)。
- 執行目標檔案，等待結束，或是等待一段時間。
- 執行後，除了把虛擬機器的記憶體再次記錄下來外，也在 GUEST OS 內使用 USER-MODE 指令讀取通訊阜開啟列表。
- 讀取兩個記憶體映像檔的內容，並且找出通訊阜開啟列表。
- 比對兩個列表，找出這段時間內新增的通訊阜位置。

比對新增的通訊阜位置與 User-mode 指令所讀取通訊阜開啟列表，找出隱藏的已開啟通訊阜。

偵測驅動程式植入行為：為找出程式是否會動態載入可疑的驅動程式，需要比對不同時間先後的載入列表。列出系統中所有被使用的驅動程式，透過篩選出最近新增的，將可把可疑的惡意驅動程式找出。惡意驅動程式往往是 rootkit 為了有更高的執行權限而慣用的手法。Rootkit 進入系統時，大多會存取檔案系統，或者寫入一些資訊到硬碟裡。伴隨著惡意程式，可能會有自我複製、修改系統檔案等。當使用者查覺有可疑檔案時，欲刪除或還原的時候，可能因為 rootkit 佔住執行權而不讓使用者刪除或是修改。有些 Rootkit 程式為了隱藏這些檔案，會使用 API Hooking 的方式，干擾對硬碟的查詢動作，讓使用者完全無法由正常的管道得知這些檔案的存在。一般來說，在 Windows OS 下要取得檔案列表，可以透過 Win 32 函式庫 FindFileFirst 或 FindFileNext。因此，若 rootkit 想達到隱藏檔案的目的，就會攔截（hook）這

兩個函式呼叫，如果回傳的檔案名稱是 rootkit 想要隱藏的，就略過這個檔案名稱，讓該檔案無法顯示出來。如此一來，使用者就無法察覺有一隱藏檔案存在，而防毒軟體也可能透過這兩個 API 去取得檔案列表掃毒，所以也無法針對被 rootkit 隱藏的檔案進行掃毒。除了對 Hook Win32 API 外，也可以利用 Windows Driver 的階層式架構來達成隱藏檔案的目的(如下圖)。 此類型的 Rootkit 會把自己註冊一個驅動程式，當一個開啟檔案的要求送往檔案系統的時候，Rootkit 將會攔截到並且判斷此要求是否為 Rootkit 想要隱藏的檔案。如果是，Rootkit 將回傳開檔失敗訊息或是將要求導到別的檔案藉此欺騙使用者或防毒軟體。當一個寫檔的 I/O 進來時，Rootkit 可以竄改寫檔的內容或是改變讀檔的結果。

以下將透過正規化的方式，描述本平台的演算法以及各元件間的關係。

名詞	**定義**
ALLURL(u)	A set of links that can be traversed by a URL u.
File(u)	A file downloaded by network crawler with URL u
UUID(f)	An universally unique identifier of file f
DB(uuid)	Query a File f with UUID uuid in database
DB_store(uuid , f)	Store a File f with UUID uuid in database
DB_report(uuid , r)	Store a Report r with UUID uuid in database
Q	A global queue stores UUIDs wiht actions push() and pop()
Analyze()	Action taken by analyzer and generate report
Analyzer	A set of analyzers used in this system
TBA	Taint-Based Malware Behavior Analyzer
ClamAV, Kaspersky	Anti-viurs software

關聯性
Given a URL u, if uuid = UUID(File(u)) , then File(u) = DB(uuid).
{TBA, ClamAV, Kaspersky } ⊂ Analyzer .

檔案來源子系統 與 儲存子系統
Given a URL u as an input to network crawler. For each URL $u_i \in$ ALLURL(u) F ← File(u_i) uuid ← UUID(F) DB_store(uuid , f) push(uuid) end

儲存子系統 與 分析器管理子系統

For each Analyzer A_i in our system
while(uuid $\leftarrow$ pop())
F $\leftarrow$ DB(uuid).
#A_i start analyzing and generate report
r $\leftarrow$ Analyze()
DB_report(uuid , r)
end

貳、計畫項目完成度

本計畫於期兩年共實作下列五項工作，完成度皆為 100%：

一）網路檔案蒐集(完成度：100%)

此模組會藉由網域種子清單，來當作擷取網站檔案模組的起點。透過瀏覽這些網站的網頁，能夠找出對外連結的 URL，以擴充探索的網域範圍。在網站回覆的網頁 (html, htm) 裡，將會有檔案的下載連結。再透過這些連結，網路爬蟲即可擷取這些放置在網站的檔案。本系統並非只是單純的下載檔案，而是透過一層一層的網路連結關係，擴大搜索整個網路上的檔案，與搜尋引擎類似。而每擷取到一個 URL，將會將檔案下載，除放置分散式資料庫中。本項目利用現有的搜尋引擎客製化系統 Nutch，來達到上述之功能。Nutch 是一款基於 Hadoop 時實作出的網路搜尋引擎，該系統可以爬取網路上的資料，透過分析建檔來提供資料搜索的服務。但本模組並不需要後續對網頁分析的部分，並且有實作與後端資料庫存取的部分。

二）分散式的資料儲存(完成度：100%)

NoSQL 資料庫提供了 Key-Value 的存取方式，利用特殊的雜湊函式來保證讀取和寫入為常數時間。本子系統將結合分散式資料庫 Cassandra 與集中式 Key-Value Redis 來實現分散式分析資料儲存。由於 Cassandra 是分散式的資料庫，在存取的時間與存取的能力較傳統 Sql 系統來的有優勢。但是其缺點是無法快速地列舉其儲存的資料。為了依序處理分析平台內待處理的分析工作，我們使用了 Redis 內所提供的 List 資料結構來實現佇列的功能，依序地從工作佇列取出欲分析的工作識別（tasd ID）。在按照取出的工作識別去 Cassandra 資料庫內尋找更多的檔案資訊。

三）惡意程式分析(完成度：100%)

本子系統將分析程式模組化，藉由著制定的介面來達到有彈性的掛載分析工具。由於分析工具可能會有數種，需要統一化其分析工具的 I/O。提供一套組態設定來讓此分析平台可以掛載和移除分析模組。對於本系統與分析模組的銜接部分，將會定義各種分析工具的 I/O 要如何導入與輸出資料，藉著一套統一的機制來方便管理不同實作出的分析工具，來達到本分析平台可擴充的能力，讓使用者可以新增自己的分析工具，讓本平台的分析功能多樣化。

四）文件 Call/Pop 分析器(完成度：100%)

本分析模組會檢查可執行檔是否具有堆疊位址取得的技術。給予一個欲檢測的執行檔，本分析模組能夠判斷該程式是否會刻意的存取堆疊的內容。如果該執行檔試圖取出堆疊的內容，將會被此系統偵測出。此分析模組會包含一個模擬的 x86 CPU，來模擬堆疊與 CPU 暫存器的狀況。而我們透過這 CPU 模擬器，可以完全觀測此程式會不會取出事先寫入在記憶體中的魔法數值(magic number)。不同於虛擬機器，此分析模組不用模擬所有的硬體裝置，只要簡單的模擬 CPU 暫存器與堆疊，故能減少其他較不必要的運算開銷。由於 CPU 模擬器的實作較複雜

冗長，我們將會結合利用 libemu 來縮短此分析模組的開發時間。藉由著修改 libemu 的系統，觀測模擬運算時記憶體的狀態，來實作出偵測取得堆疊位址資訊的可疑檔案。

五）全系統層次的行為分析系統（完成度：100%）

本分析系統將利用動態汙染源技術來分析惡意程式在虛擬機器上的攻擊行為。藉著整合汙染源追蹤系統來觀測資訊流動的狀態，來判斷該程式之行為。由於資訊流動都是以機械碼為基礎來觀測，故惡意程式的行為可以鉅細靡底的被列出。由於這些二進位的資料，對於一般安全人員並無太大的意義。所以，本全系統層次的行為分析系統，將會把這些觀測到的資訊流動還原到作業系統中對應的物件存取，以利判斷分析。該系統將包含下列：偵測程序修改行為(PRC)、偵測檔案修改行為(FLM)、偵測登錄機碼修改行為(REG)、偵測網路通訊修改行為(NET)、偵測驅動程式植入行為(DRV)。

參、計畫成果

本計畫提出一個新全系統層次的行為分析技術來擷取惡意程式行為並將其整合至雲端分析平台中，詳細方法已於第壹章第四節研究方法中說明，依據該方法我們已經開發完成系統，在第本章第五節系統建置將詳細介紹，其內容有實際佈建情況、系統架構、工作流程、實作細節以及實驗數據結果。同時，本計畫並且獲得多項成果，包含產學合作計畫、學術貢獻等部分，將在以下各節中詳述。

本計畫的雲端惡意程式分析平台包含下列四項關鍵技術：１）雲端節點機器管理、２）分析工作分配與排程、３）可擴充式掛載介面、４）動態汙染源技術，將會在第肆章節技術優越性詳細敘述，並於第伍章寫明本計畫的結論與未來展望。

一、產學合作計畫

本計畫所涵蓋的研究範圍以及所發展的技術深具產業應用價值，已陸續與友訊科技與中華電信等國內業界大廠簽訂產學合作計畫，其計畫名稱與合作方式如下圖表 6：

合作計畫名稱	合作對象
動態惡意程式行為側錄與污染分析	中華電信
DNSsec 推動先期型計畫	教育部
DNSSEC 網域名稱安全架構建置與推廣計畫	教育部
委託 Open D-Link Routers Forum 建置、維護與測試技術與諮詢服務	友訊科技
法務部調查局惡意程式自動檢測技術支援系統委託研究案	調查局
前瞻性檔案完整性驗證與可疑嵌入碼檢測平台	喬鼎科技
雲端行動的安全及時分析可行性評估先期探討	工業技術研究院
智慧終端技術研究	工業技術研究院
雲端惡意程式鑑識與行動平台安全	宏達電子(HTC)
雙階層式全系統汙染鑑識分析	中華電信
Technology Transfer on Network Threat Detection using Security Log Correlation	趨勢科技

圖表 6 產學合作計畫總表

二、 學術貢獻

- **期刊論文:**

1. Chia-Wei Hsu, C.W. Wang, Shiuhpyng Shieh, “Reliability and Security of Large Scale Data Storage in Cloud Computing,” IEEE ATR, 2011.
2. Pokai Chen, Shiuhpyng Shieh, “Security for Future Internet Architecture - Motivation from DNSSEC,” IEEE ATR, 2011.
3. C.W. Wang, Shiuhpyng Shieh, “SWIFT: Decoupling System-Wide Information Flow Tracking for Malware Analysis,” in revision, IEEE Transactions on Reliability.
4. LY, Yeh, Y.L. Huang, S.P. Shieh, Anthony Joseph, “A Batch Authenticated and Key Agreement Framework for P2P-based Online Social Networks,” IEEE Transactions on Vehicular Technology, 2012.
5. Yu-Lung Huang, Chiya Shen, Shiuhpyng Shieh, "S-AKA: A Provable and Secure Authentication Key Agreement Protocol for UMTS Networks," IEEE Transactions on Vehicular Technology, Volume 60, Issue 9, Nov. 2011, PP. 4509 – 4519.

- **會議論文:**

1. Yin-Change Song, Michael Cho, C.W. Wang, Chia-Wei Hsu, Shiuhpyng Winston Shieh, “Light-Weight CSRF Protection by Labeling User-Created Contents,” submitted to IEEE Symposium on Software Reliability Engineering (ISSRE), 2012.
2. Chung-Kwan Chen, Wei-Chi Chen, Vic Hsu, Shiuhyng Shieh, “Mutant Malware Discovery and Behavior Analysis for Cyber Crime Investigation,” Crypto and Information Security Conference, 2012.
3. Fan-Shin Shih, Chia-Wei Hsu, Shiuhpyng Shieh, “Detecting VM-Awareness Using Divergent Multi-Execution Traces,” submitted to USENIX Symposium on Operating Systems Design and Implementation, 2012.
4. Yun-Min Cheng, Bing-Han Li and Shiuhpyng Shieh, “Accelerating Taint-based Concolic Testing by Pruning Pointer Overtaint,” accepted for publication, IEEE Conference on Software Security and Reliability (SERE), Washington DC, June 2012.
5. Chia-Ming Fan, Bing-Han Li, Shiuhpyng Shieh, “On The Security of Password-Based Pairing Protocol in Bluetooth” The 13th Asia-Pacific Network Operations and Management Symposium, (APNOMS 2011), 2011
6. Yen-Ru Liu, C.W. Wang, J.W. Hsu, T.C. Tseng, S.P. Shieh, “Extracting Hidden Code from Packed Malware based on Virtual Machine Memory Comparison,” 21th Cryptology and Information Security Conference (CISC 2011), 2011.
7. Bing-Han Li, Shiuhpyng Shieh, “RELEASE: Generating Exploits Using Loop-Aware Concolic Execution,” IEEE Conference on Secure Software Integration and Reliability Improvement, June 2011

8. Wei Shi-Sue, Shiuhpyng Shieh, Bing-Han Li, Michael Cheng Yi Cho and Chin-Wei Tien, "A Framework Using Fingerprinting for Signal Overlapping-Based Method in WLAN," in Proceedings of the International Computer Symposium on Computer Networks and Web Technologies (ICS 2010), Tainan, Taiwan, Dec. 16-18, 2010.
9. Yen-Ru Liu, C.W. Wang, J.W. Hsu, T.C. Tseng, S.P. Shieh, "Extracting Hidden Code from Packed Malware based on Virtual Machine Memory Comparison," 21th Cryptology and Information Security Conference (CISC 2011), 2011.
10. Chia-Ming Fan, Bing-Han Li, Shiuhpyng Shieh, "On The Security of Password-Based Pairing Protocol in Bluetooth," The 13th Asia-Pacific Network Operations and Management Symposium, (APNOMS 2011), 2011.

三、 產業貢獻

國外專利

1. CW Wang, SP Shieh, YR Liu, "Method for Decoupling System-Wide Information Flow Tracking for Malware Analysis and Its Applications," US patent pending
2. S.I. Huang, S.P. Shieh, "Method and System for Secure Data Aggregation in Wireless Sensor Networks," China patent number ZL200710301500.9, 2011.
3. S.I. Huang, S.P. Shieh, "Method and System for Secure Data Aggregation in Wireless Sensor Networks," US patent no. 8027474, 2011.9.27.

國內專利

1. 王繼偉，謝續平，劉晏如，"分離式的全系統層次模擬器與資訊流動追蹤方法與其應用," Taiwan patent pending
2. S.I. Huang, S.P. Shieh, "Method and System for Secure Data Aggregation in Wireless Sensor Networks, 用於在無線感應器網路中進行安全資料聚合的方法以及系統" ROC patent no. I350086. 10, 2011.

四、 特殊榮譽

學生獲獎

1. 2010 Hacks in Taiwan Conference 台灣駭客年會 Wargame 競賽冠軍
2. 2011 Hacks in Taiwan Conference 台灣駭客年會 Wargame 競賽冠軍
3. 2012 Hacks in Taiwan Conference 台灣駭客年會 Wargame 競賽殿軍
4. 2011 資策會資安技能金盾獎亞軍

主持人社會服務

1. 總統府國家安全會議顧問
2. 國家資通安全會報技服中心惡意程式交流聯盟主席

五、 系統建置-雲端惡意程式分析平台

本項目將簡介本子計畫於第一年度的系統建置情況，將分別介紹實際佈建狀況、系統架構、工作流程、系統實作、效能分析。經由以上的介紹可突顯本計畫將不只著力於新興技術的研究，更注重實務的系統開發。

■ 實際佈建

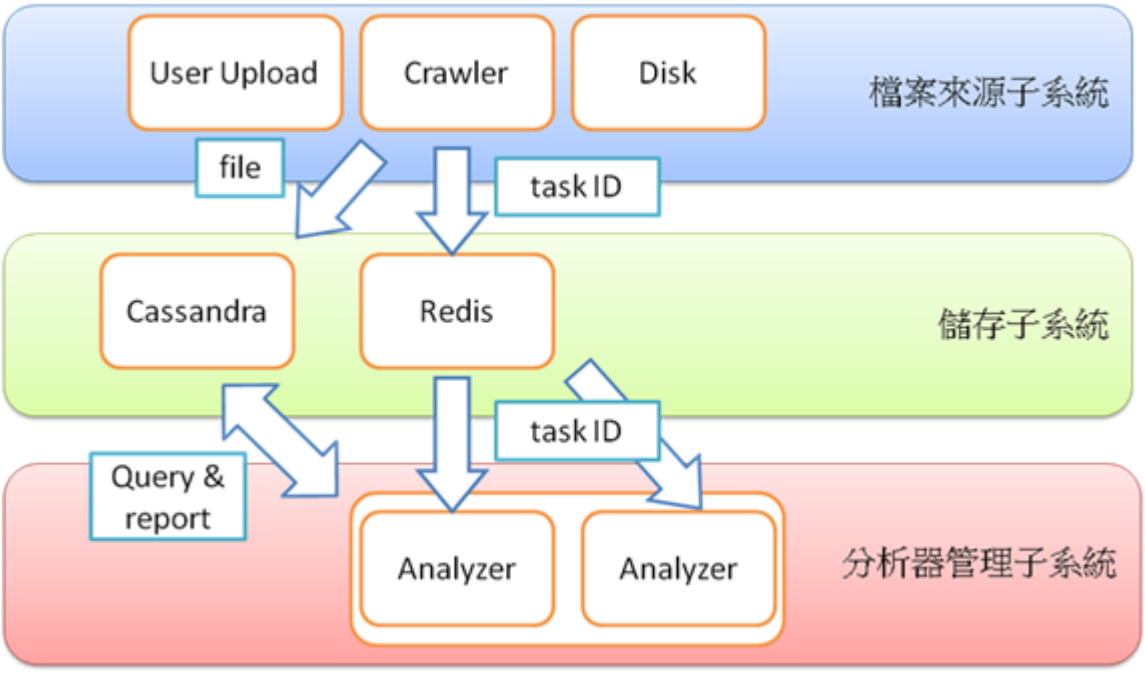
本子計畫於第一年度100年實際架設Windows惡意程式行為分析雲端平台，並建置於國立交通大學網路安全實驗室內，以進行第一年度的雲端分析平台研究。本平台利用市面上能夠採買之高效能運算個人電腦，以節省開發成本。利用這些高運算量來實現「基於虛擬機器技術之動態程式行為分析」以及「高可靠性惡意程式樣本雲端儲存系統」。在動態程式行為分析部分，透過整合本實驗室所開發之惡意程式分析模組，有效地利用此平台的運算效能。而雲端樣本儲存系統，則可以大量的儲存已知或是分析過可疑的惡意樣本，以供往後的研究、查詢。此平台已經透過產學合作之關係，正與調查局、中華電信、趨勢科技、交通大學資訊服務中心等相關產業合作。進行實務性質的惡意程式分析，以提供多樣化的病毒分析研究。

圖表 7 WINDOWS 惡意程式分析雲端平台實際佈建

圖表 8 網頁資料爬取子系統，可蒐集網頁上的資料進行分析

■ **系統架構**

本分析平台主要由三個部分所組成：一）檔案來源子系統、二）儲存子系統、三）分析器管理子系統。在檔案來源子系統中，本子系統需要接收大量的檔案作為分析的對象，以分析並取得各種不同型態的惡意軟體。本系統實做一個 PushTask 之函式庫，此 API 可將任意檔案輸入本系統進行分析，並指定該檔案的優先權。此 PushTask 函式庫先產生檔案的工作識別（UID），在將此 UID 放入 Redis 中，並將檔案內容存入 Cassandra 中，將檔案輸入我們的系統中。利用此 API 為基礎，我們實做了兩種檔案來源系統：網頁爬取系統和批次檔案輸入。在儲存系統方面，由於此研究已經被探討多年，市面上也有大量的資料庫可以應用於本分析平台上。為了節省開發時間以及維護方便，經過深入地了解以及搜索之後，我們選擇了兩個現有的資料庫系統 Cassandra 和 Redis，透過修改程式、設定檔案和內部些許的程式碼，快速地達到本分析平台所需要的儲存能力。Cassandra 是一套分散式的資料庫系統，在分析平台中用來儲存未分析的檔案內容以及分析完的結果。Redis 是一單機的資料庫系統。在分析平台中以佇列(Queue)的模式來儲存未分析的檔案的工作識別號碼。在分析器管理子系統中，藉著可以掛載多個分析器的能力。同時進行多個分析器的分析，以取得更準確的結果。因此我們需要建構一套分析器管理子系統來管理各種不同的分析器，讓各種不同輸入輸出的分析器能正常運作於本系統中。



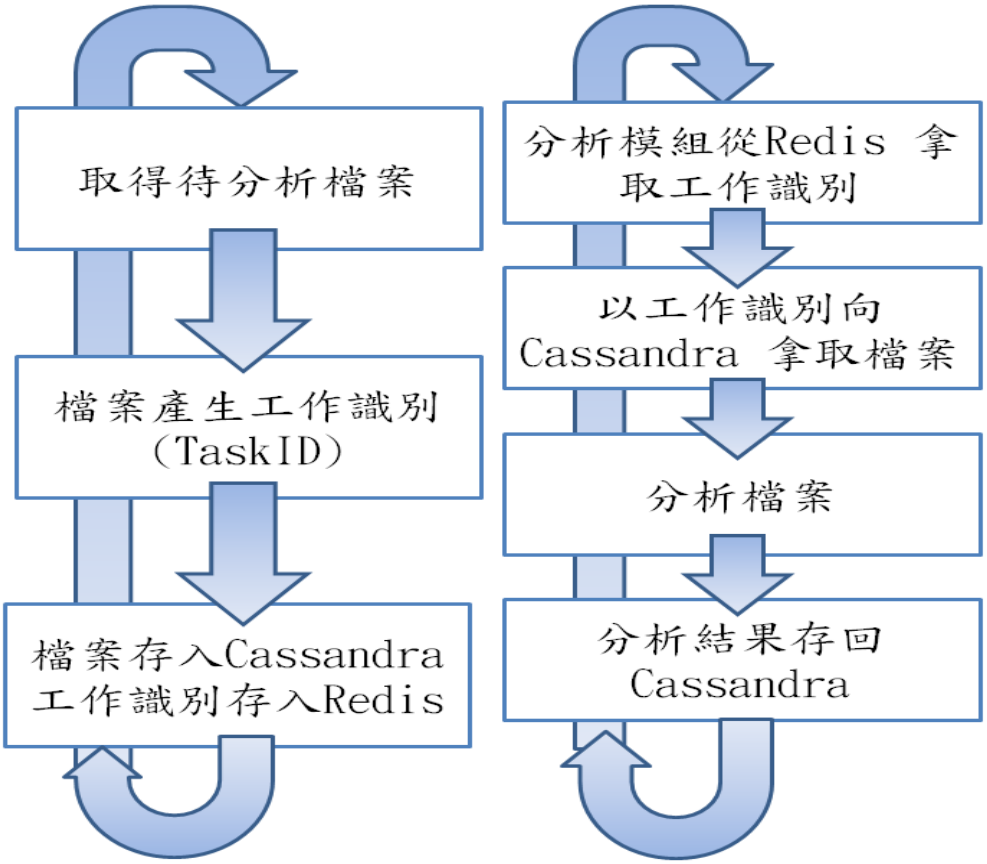
圖表 9　基於機器碼之 Windows 惡意程式行為分析雲端平台系統架構圖

■ **工作流程**

本系統主要的流程可分為兩部分(如圖表 14)，彼此間連繫的管道是透過儲存子系統，詳細如下：

1.由檔案來源子系統至儲存子系統:負責將檔案存入系統中，同時產生後續分析所需的工作識別。此處的檔案來源目前有三種:使用者上傳、硬碟讀取和Crawler。

2. 由儲存子系統至分析器管理子系統:負責處理未分析的檔案。首先至儲存子系統中的Redis拿取工作識別後，再去向Cassandra拿取檔案來進行分析，之後儲存結果。



圖表 10　　基於機器碼之 Windows 惡意程式行為分析雲端平台工作流程

■ **系統實作概要**

以下將會詳細地列出實作的細節，藉著閃示其虛擬碼，讓讀者能更清楚的了解各部分的流程。

程式 名稱	功能	參考 pseudocode
secmap.rb	啟動本雲端惡意程式分析平台	**圖表 15**
startnutch.sh	啟動網路爬取子模組	**圖表 16**
NutchSlave.rb	傳送爬取網頁內容至分析平台	**圖表 17**
AnalyzerInvoker.rb	呼叫分析器子模組	**圖表 20**
common.rb	環境變數的設定和產生工作識別號碼（TaskID）	**圖表 23**
invokeAnalysis.rb	啟動分析器，之後拿取未分析檔案進行分析。	**圖表 21**
putToRedis.rb	Analysis input 元件將取得檔案的 TaskID 存入 Redis，並且依據不同的檔案來源給予不同的優先權。	**圖表 18**
putToCassandra.rb	Analysis input 元件將檔案放入 Cassandra。	**圖表 19**
getTaskID.rb	invokeAnalysis.rb 取得存放在 Redis 內的 TaskID。	**圖表 22**
getFileContent.rb	invokeAnalysis.rb 用 TaskID 至 Cassandra 取得相對應的檔案。	**圖表 24**
saveReportToCassandra.rb	分析器將分析結果存入 Cassandra。	**圖表 26**
getReportFromCassandra.rb	將分析器分析完的結果取出。	**圖表 25**

- Usage:
  - ./secmap [start|stop] [cassandra | dispatcher | nutch | ANALYZER_NAME]
- EX:
  - ./secmap start cassandra mba-taint-1 clamav

```
$starttable = {
"cassandra"=> \
"./cassandra/bin/cassandra",
"dispatcher" => \
"./dispatcher.rb",
"nutch" => \
"./startnutch.sh" }

$stoptable = {
"cassandra"=> \
"kill -9 `cat cassandra.pid`",
"dispatcher" => \
"kill -9 `dispatcher.pid`",
"nutch" => \
"kill -9 `nutch.pid" }
```

```
secmap ( cmd , roles[] ){
    if( cmd == "start")
       foreach role in roles
           `$starttable[role]`
    else if( cmd == "stop")
       foreach role in roles
           `$stoptable[role]`
    else
       do_error_msg();
}
```

圖表 11 SECMAP.RB 之虛擬碼

本平台的運行主要下的指令如圖表 15，我們將所有系統指令包裝成一個執行檔案。並且會有許多個元件，可以互相的配合執行。圖表 15 的./secmap 是整合本系統個元件主要執行程式，透過此介面可以快速啟動、停止下列四個系統主要元件(role)：1.)cassandra，本系統中用來儲存大量資料的分散式資料庫、 2.)dispatcher，用於儲存工作與分配的佇列、3.)nutch，為本系統來爬取網路資料的元件及 4.)各種 analyzer，掛載於本系統的不同分析器。starttable 和 stoptable 是兩個 Hash 資料結構，將系統元件的名稱對應到一字串值。此介面利用 starttable 將啟動系統儲存及網路爬取元件的所需執行的動作儲存，並利用 stoptable 將停止系統儲存及網路爬取元件的動作儲存，並根據所取得的參數執行相對應的動作。透過使用 starttable 和 stoptable，可以同時啟動多個不同的元件。例如: ./secmap start cassandra redis，便可以一次將 cassandra 和 redis 運行起來。

以下將對上圖做詳述:

(一) 參數介紹：

cmd：代表所要執行的動作，分為 start 和 stop 兩種指令。

roles[] ：執行指令的目標元件的陣列，共有 cassandra、nutch、dispatcher、analyzer(以名稱區別)

(二) 判別執行指令是’start’或是’stop’。

(三) 對 roles 裡面的各個元件，根據所要執行的指令(stop 或 start)，查詢並執行其對應(starttable 或 stoptable)的指令。

```
startNutch (void) {
    nutch_inject_seeds();
    #bin/nutch inject $dir/crawldb seed
    while(1){
        nutch_generate_URLs();
        #bin/nutch generate $dir/crawldb/ $dir/segments/ -top 100000 -numFetchers 100;
        nutch_get_download_filename();
        #s=`bin/hadoop fs -ls $dir/segments/ | cut  -d " " -f 19 | tail -n 1`;
        nutch_fetch_files();
        #bin/nutch fetch $s -threads 100 ;
        nutch_update_URLs();
        #bin/nutch updatedb $dir/crawldb $s ;
    }
}
```

startnutch.sh

Nutch slaves call ***NutchSlave.rb***

圖表 12 STARTNUTCH.SH 之虛擬碼

startnutch.sh 負責在主節點將 nutch crawler 啟動並透過 map-reduce 的機制將爬取網頁的工作自動化分配至各附屬節點上。由於 nutch 原本提供的爬網功能包含許多我們系統不需要的部分，如：製作索引、建構搜尋網頁，因此我們直接利用 nutch 提供的 API 執行爬網的功能，能提高 nutch 爬網的效率，增加每天搜尋過的網頁數量。

以下將對圖表 16 做詳述:

(一)參數介紹：

Seed：為一包含多個網址檔案，代表系統欲抓取的起始網頁(seed)。

(二) nutch_inject_seed

nutch_inject_seed 負責將這些網頁輸入待爬取資料庫(crawldb)，以便 nutch 之後由這些網頁開始爬取網頁。圖()是 seed 檔案的範例。

```
http://seclists.org/bugtraq/
http://zeltser.com/combating-malicious-software/malware-sample-sources.html
http://contagiodump.blogspot.com/2010/11/links-and-resources-for-malware-samples
.html
http://www.malware.com.br/
```

(三) nutch_generate_URLs、nutch_get_download_filename

此函數由帶爬取資料庫中選取前若干組網頁資料，在本系統中預設是抓取 100,000 組網頁資料，產生一組待爬取清單(fetch list)。並將產生的清單名稱傳給 nutch_fetch_files 作為輸入。

(四) nutch_fetch_files

根據取得的待爬取清單，利用 mapreduce 將檔案分配至各台 slave 上面做爬取的動作。而個別 slave 在抓取到可執行檔後，便會執行 NutchSlave.rb 將檔案儲存到資料庫中。

(五) nutch_update_URLs

將在 nutch_fetch_files 找出來的新連結加入待爬取資料庫，並更新已爬取網頁的資訊

NutchSlaveCrawler.java

```
class NutchSlaveCrawler{
        public NutchSlaveCrawler(String filename){
          Runtime rt = Runtime.getRuntime();
          Process proc = rt.exec("NutchSlave.rb "+ filename);
        }
}
```

NutchSlave.rb

```
NutchSlave ( filename ){
    uid = generateSecmapUID( filename );
    commandsTable = loadCommandsTable();
    result = `$commandsTable['putToCassandra'] UID:${filename}`
    if( result )
        error_handle();
    result = `$commandsTable['putToRedis'] UID:${uid} priority:2`
    if( result )
        error_handle();
}
```

圖表 13 NUTCH 之原始碼物件以及 NUTCHSLAVE.RB

當我們在主節點執行 nutch 後，主節點(nutch master)便會自動將所需執行的工作分配至附屬節點(slave)做網頁爬取的動作。我們改寫 nutch 的程式碼，在做網頁爬取的部分我們加入了 NutchSlaveCrawler()，此函數利用 JAVA 提供的 Runtime 及 Process 函式庫去執行 NutchSlave.rb，並將爬取下來的網頁檔名做為其參數。NutchSlave.rb 負責將檔案上傳至我們系統。

以下將對圖表 17(NutchSlave.rb)做詳述:

(一)參數介紹：

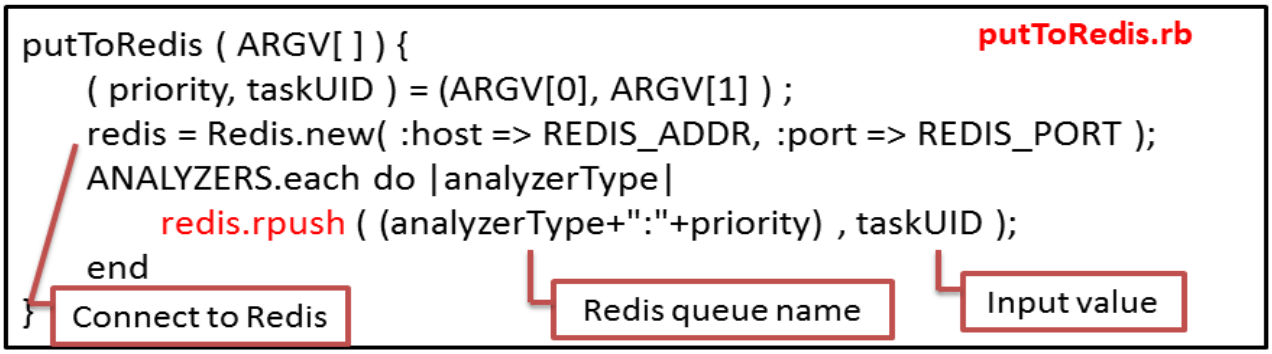
filename：Nutch 所抓取下來的檔案名稱。

(二) 首先呼叫 generateSecmapUID，計算出檔案的 MD5 以及 SHA1 值，再加上檔案本身大小作為其唯一識別(UID)回傳。

(三) 接著利用 loadCommandsTable()取得各個函式庫的位置。

(四) 透過 commandsTable，NutchSlave.rb 先執行 putToCassandra 將檔案內容及屬性存入分散式惡意樣本資料庫內。

(五) 最後利用 putToRedis 將檔案的唯一識別輸入特定優先權的工作佇列，等待分析器處理。



圖表 14 putToRedis.rb 之虛擬碼

此程式 putToRedis.rb 是用於將 taskUID 根據不同的優先順序(priority)存入 Redis Queue 中。另外，本系統針對不同的分析器都產生一組 Redis Queue(針對不同的 priority)。EX: 有分析器 A 和分析器 B，兩種分析器，並同時有 3 種優先序，因此一共有 6 個 Redis Queue。以下將對圖表 18 做詳述:

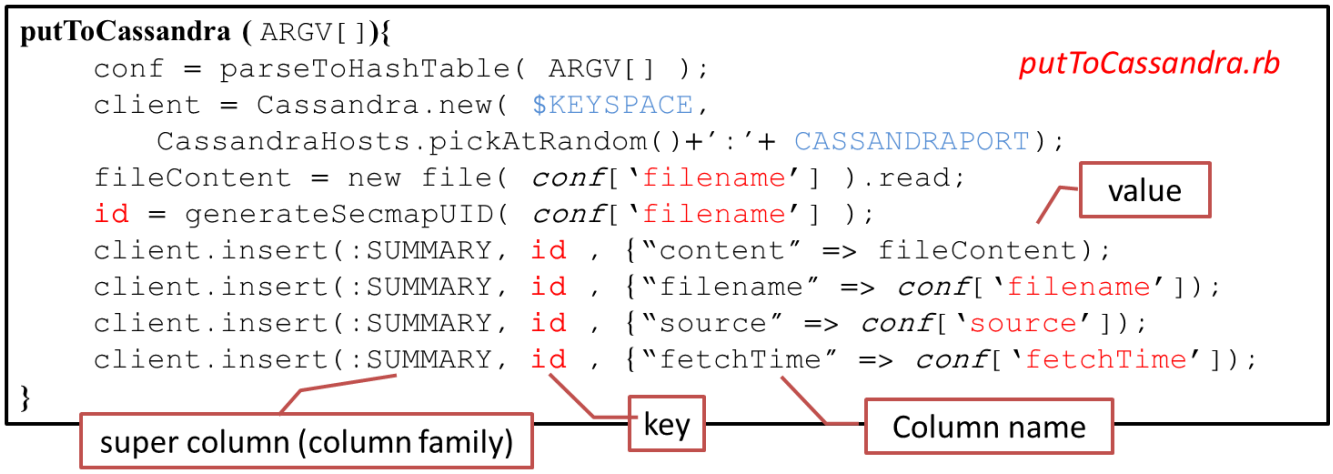
(一)參數介紹:

priority: 代表這筆資料的優先序，同時也是 Redis Queue 命名的一部分。

taskUID: 由分析檔案所產生的 ID，而產生方法則是將檔案內容作 MD5 以及 SHA1，並將兩者相連最後連上檔案大小而得。

(二) 連線到 Redis server，其中的變數 REDIS_ADDR 和 REDIS_PORT 分別代表 IP 以及 port，EX:1.2.3.4 和 10001。

將此 taskUID 透過 redis.rpush 放入所有分析器相對應優先序的 Queue 中，以上述的 A B 分析器為例，當優先序為 3 時，則會將 taskUID 放入 A3 以及 B3 的 Queue 中。



圖表 15 PUTTOCASSANDRA.RB 之虛擬碼

此程式 putToCassandra.rb 的作用是將取得的檔案名稱、取得的時間、檔案的來源以及檔案內容存入資料庫中。

以下將按步驟解說，如圖表 19 所示:

(一) 首先，透過 parseToHashTable 先從 ARGV[] 內取出檔案的基本資料，有檔案名稱、來源和取得時間。

(二) 再來，連線至資料庫，其中 CassandraHosts.pickAtRandom()是用來連到不同的資料庫，回傳的值是任一資料庫的 IP 地址，目的是為了流量能夠均勻分到每一台分散式的資料庫上。

(三) 取出檔案資料。

(四) 針對檔案內容透過 generateSecmapUID()產生相對應的 ID，而產生方法則是將檔案內容作 MD5 以及 SHA1，並將兩者相連最後再連上檔案大小而得。

(五) 最後透過 client.insert 來對資料庫的相對應欄位(super column:SUMMARY 和 column name:content…etc)填入相對的值(value)。

```
AnalyzerInvoker ( analyzerName ){
    commandsTable = loadCommandsTable();
    while(1){
        taskID=`$commandsTable['getTaskID'] analyzerName:$analyzerName`
        if( taskID ) error_handle();
        result = `$commandsTable['getFileContent'] \
                                    taskID:${taskID} outfile:a.exe`
        if( result ) error_handle();

        analyzerConf = readAnalyzerConfig( analyzerName );
        result = `${analyzerConf['COMMAND']} a.exe`
        if( result ) error_handle();

        result = `$commandsTable['saveReportToCassandra'] \
                                    log:${analyzerConf['LOG']}`
        if( result ) error_handle();
    }
}
```

圖表 16 AnalyzerInvoker 之虛擬碼

AnalyzerInvoker 負責將分析器運行起來，自動從系統取得檔案並執行分析，最後再將結果存回系統。分析器只要放置於特定目錄(如：~/analysis)，並且具有 config 檔案並設定啟動分析器的方式以及報告儲存的位置，便可以整合進我們系統。以下將對圖表 20 做詳述:

(一)參數介紹：

analyzerName：所要執行的分析器名稱

AnalyzerInvoker 會先執行 loadCommandsTable 取得各個 API 的位置。

(二) 執行 readAnalyzerConfig，此函數分析分析器 config 設定，記錄分析器執行方式及報告位置。執行 getTaskID API，此 API 會連線至 redis 工作佇列，取得指定分析器所需要分析的檔案 UID。

(三) 透過檔案的 UID，我們可以利用 getFileContent 從 cassandra 取得檔案內容，並儲存於本地端。

(四) 根據分析 config 取得的執行方式，AnalyzerInvoker 可以利用 invokeAnalysis API 將分析器執行起來，並將待分析檔案傳給分析器做為參數，

(五) 最後呼叫 saveReportToCassandra 將分析報告儲存回 Cassandra 資料庫。

```
command = getCommand()
filename = getFilename()
analyzerPid = fork{
    Signal.trap("SIGXCPU") do
        timeout()
    end
    Process.setrlimit( Process::RLIMIT_CPU , CLEAN_UP_TIME, FORCE_QUIT_TIME )
    `#{command}`
}
Process.wait(analyzerPid)
```

圖表 17 INVOKEANALYSIS.RB 之虛擬碼

invokeAnalysis.rb 會接收啟動分析器所需的命令以及待分析的檔案名稱作為參數，並 fork 出一支新的行程並啟動分析器以分析檔案。

以下將對圖表 21 做詳述:

(一)參數介紹：

command：執行分析器所要執行的指令

filename：所要分析的檔案名稱

(二) invokeAnalysis.rb 執行 getCommand()、getFilename()從參數列取得分析器執行指令以及待測檔案名稱。

(三) 接著利用 fork()指令創造出一支新的行程執行分析。

(四) 為了避免分析器運作太久，系統必須監控分析器執行狀態，並在分析器超時後中斷分析器的執行。因此我們利用 setrlimit()設定 CPU time 的限制。

(五) 當行程執行超時候，作業系統會自動發出 SIGXCPU 訊號並結處該行程，SIGXCPU 是當一個行程資源使用量超過系統規範實，用來通知該行程的訊號。我們利用 Singnal.trap()去抓取 SIGXCPU 訊號，並執行 timeout()紀錄分析器運行的狀況。

(六) 執行 command 將分析器運行起來。

```
getTaskID ( ARGV[] ){                                          getTaskID.rb
    conf = parseToHashTable( ARGV[] );
    redis = new Redis (:host => REDIS_SERVER, :port => REDIS_PORT);
    for i ← 0 to LOWEST_PRIORITY {
        if( taskID = redis.lpop(conf['analyzerName']+i.to_s) != NULL )
            break;
    }
    if( taskID == NULL ) printf_exit("all task done!");
    return taskID;
}
```

圖表 18 GETTASKID.RB 之虛擬碼

此程式 getTaskID.rb 是從 Redis 中取出一組 TaskID。其中，Redis 為一 Queue 的資料結構，在此基礎上實作出優先序(priority)的功能;而 TaskID 則是一組用來查詢資料庫的 key，key 的產生法法於 putToCassandra.rb 處有說明。

以下將對上圖做詳述:

(一)首先，透過 parseToHashTable 先從 ARGV[] 內取出分析器名稱(analyzerName)。

(二) 連線到 Redis server，其中的變數 REDIS_ADDR 和 REDIS_PORT 分別代表 IP 以及 port，EX:1.2.3.4 和 10001。

(三) 在 For 迴圈內部則是藉由 redis.lpop()來內取出 Redis 中的一筆資料，從優先序高的先取，取得後離開迴圈。

最後結果，回傳 taskID 或是提示 Redis 內已無 taskID 並顯示工作完畢。

```
ENV['ANALYZER_HOME']    = "/home/dsns/analyzers"
ENV['SECMAP_HOME']      = "/home/dsns/secmap-run"
REDIS_ADDR                  = "192.168.100.109"
KEYSPACE       = "SECMAP"
CASSANDRA      = ["192.168.100.111", "192.168.100.112", "192.168.100.113"]
CLEAN_UP_TIME     = 420 # 7 mins for clean up time
FORCE_QUIT_TIME   = 600 # 10 mins force kill analyzer

def loadCommandTable()
       rbList = `ls #{LIB_HOME} | grep .rb `
       rbList.each do | command |
             $commands[command[0..-5]] = "#{LIB_HOME}/#{command[0..-2]}"
       end
end

def generateSecmapUID( filename )
   content = File.new( filename ).read
   id = MD5.hexdigest(content)
   id <<  Digest::SHA1.hexdigest(content)
   id << File.size?(filename).to_s
    return id
end
```

圖表 19 COMMON.RB 之內容

common.rb 是此系統共用的設定及函式庫，所有相關的程式都必須先載入 common.rb。主要分成三個部分：

第一部分是本地端設定，定義系統的環境變數，這些設定在每個節點上不一定相同，因此沒個節點需要單獨設定，包含系統相關路徑與分析器路徑。

第二部分是整體系統設定，設定一些系統共用的設定，這些設定在所有節點上需要保持一致。例如：cassandra 節點網路位置、redis 節點網路位置。

第三部分是公用函式，將不同程式會使用的函式集中，方便管理及使用。包括載入指令表的 loadCommandsTable 及產生系統識別碼的 generateSecmapUID 等。

```
getFileContent (ARGV[] ){                                    getFileContent.rb
    client = Cassandra.new( KEYSPACE,
      CassandraHosts.pickAtRandom()+':'+ CASSANDRAPORT);
    vaule = client.get(:SUMMARY, $conf['taskID']);
    analyzerPath = ANALYZER_HOME+"/"+$conf['analyzerName']
    file = File.new(${analyzerPath}+"/"+$conf['outfile'],'w');
    file.write(value['content']);
}
```

圖表 20 GETFILECONTENT.RB 之虛擬碼

此程式 getFileContent.rb 是連線到資料庫端，並透過 taskID 來取得檔案，最後將檔案存在指定路徑。

以下將對上圖做詳述:

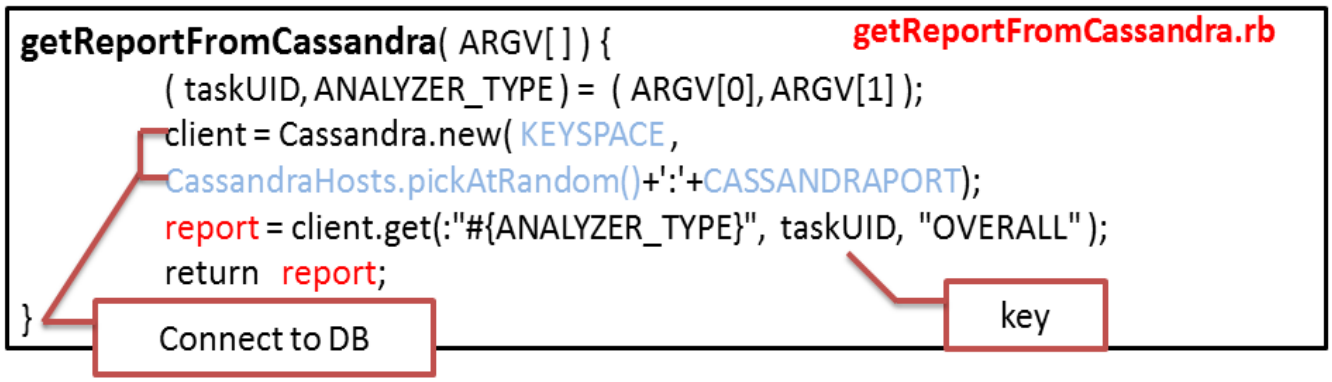
(一) 首先，連線至資料庫，其中 CassandraHosts.pickAtRandom()是用來連到不同的資料庫，回傳的值是任一資料庫的 IP 地址，目的是為了流量能夠均勻分到每一台分散式的資料庫上。

(二) 透過 client.get()函式以及 taskID 從資料庫中取得相對應的檔案。

(三) 設定分析器的路徑，由 ANALYZER_HOME 和 analyzerName 組成，之後儲存檔案的資料夾就在此路徑下。

(四) 設定檔案的儲存路徑，由 analyzerPath 和 outfile 組成。

(五) 將檔案寫入(四)的路徑中。



圖表 21 GETREPORTFROMCASSANDRA.RB 之虛擬碼

此程式getReportFromCassandra.rb 是藉由taskUID 和 ANALYZER_TYPE 來從資料庫中取得相對應的報告。

以下將對圖表 25 做詳述:

(一)參數介紹:

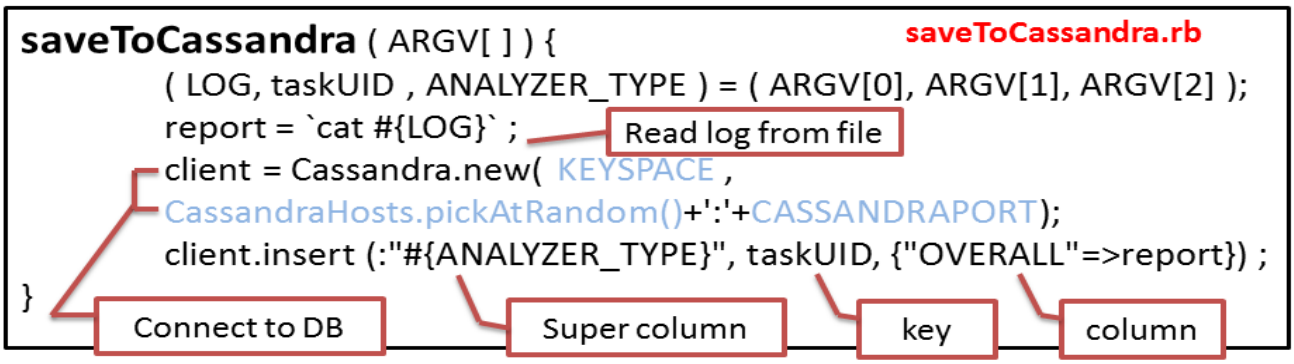
taskUID: 由分析檔案產生的 ID，產生方法則是將檔案內容作 MD5 以及 SHA1，並將兩者相連最後再連上檔案大小而得。

ANALYZER_TYPE: 分析器的種類名稱，依據此參數來區別報告。

(二) 連線至資料庫，其中 CassandraHosts.pickAtRandom()是用來連到不同的資料庫，回傳的值是任一資料庫的 IP 地址，目的是為了流量能夠均勻分到每一台分散式的資料庫上。

(三) 透過 client.get()再加上 taskUID 以及 ANALYZER_TYPE 來取得相對應的報告。

(四) 回傳報告的結果。



圖表 22 SAVETOCASSANDRA.RB 之虛擬碼

此程式 saveToCassandra.rb 是將 Log(分析過後的報告)放進資料庫中。

以下將對圖表 26 做詳述:

(一)參數介紹:

LOG: 要存入的 log 的路徑。

taskUID: 由分析檔案產生的 ID,產生方法則是將檔案內容作 MD5 以及 SHA1,並將兩者相連最後再連上檔案大小而得。

ANALYZER_TYPE: 分析器的種類名稱,依據此參數來區別報告的儲存。

(二) 首先,讀取要儲存的報告,透過 linux 內建指令 cat 來讀取報告。

(三) 連線至資料庫,其中 CassandraHosts.pickAtRandom()是用來連到不同的資料庫,回傳的值是任一資料庫的 IP 地址,目的是為了流量能夠均勻分到每一台分散式的資料庫上。

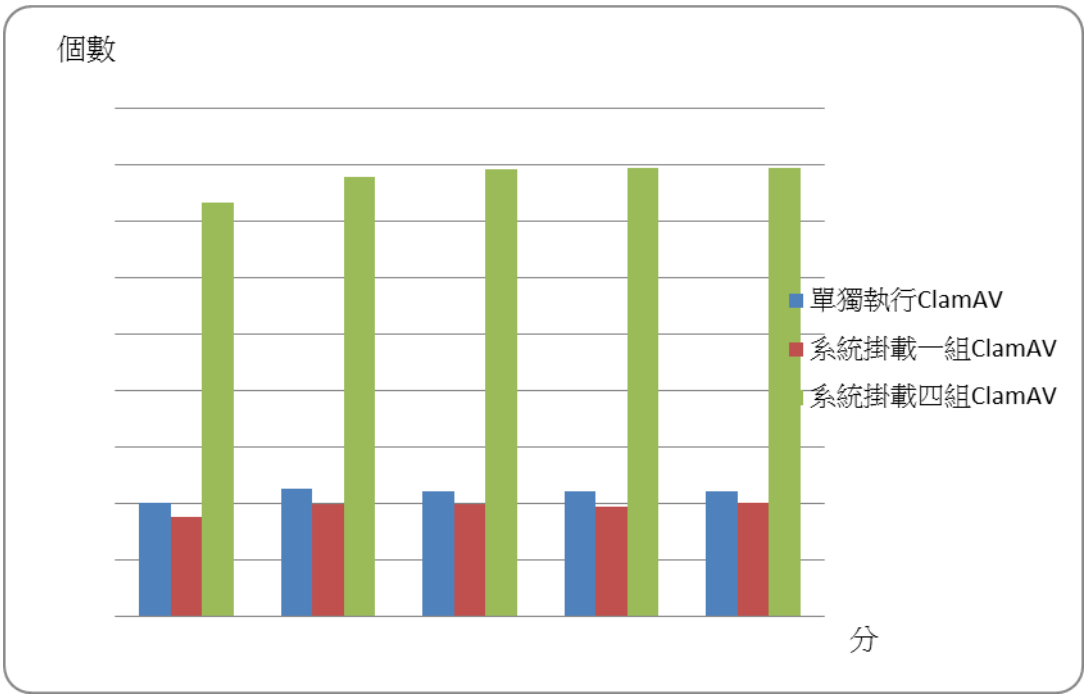
將報告存入資料庫中。透過 client.insert()來對資料庫的相對應欄位(super column:ANALYZER_TYPE 和 column name:OVERALL)填入相對的值(value)report 而 taskUID 則是 key。

■ 效能分析

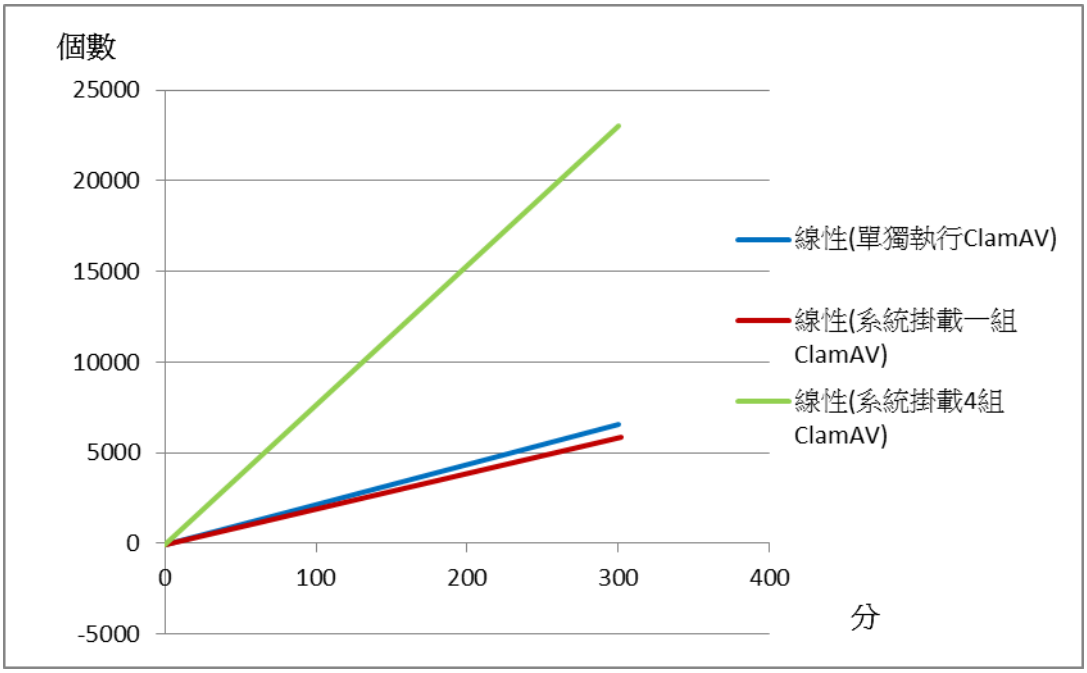
為了確認本系統的效能及可行性，我們設計了一套實驗。測試此系統在不同配置下實際運行的狀態，並與單獨分析器做比較。在此實驗中，系統採用的分析器皆 ClamAV。ClamAV 為一套開放原始碼的防毒軟體，用來偵測惡意軟體，普遍用於郵件伺服器。實驗所使用的機器皆使用 Intel(R) Core(TM) i7 CPU 搭配 16G 的記憶體。在此實驗中，本系統分別掛載執行一組 ClamAV 及四組 ClamAV，並與單獨執行 ClamAV 相比較。實驗分析目標為我們預先收集的惡意軟體樣本約 22000 隻，主要以 Windows 的可能執行檔(exe)和動態連結函式庫為主(dll)。

如圖表 28，此次實驗先以單獨執行 ClamAV 分析樣本，共需 992 分鐘處理完 22,000 個檔案。與單獨執行 ClamAV 相比較，本系統在掛載單一組 ClamAV 時，共需要 1,110 分鐘處理完 22,000 個檔案，整體效率約為單獨執行 ClamAV 的 0.89 倍。工作分配及結果儲存約種體時間的 10%，對系統負擔不大。而本系統掛載四組 ClamAV 時，只需花 282 分鐘，便可處理完全部 22,000 個檔案，整體效率為單獨執行 ClamAV 的 3.5 倍、系統掛在單一組 ClamAV 的 3.9 倍，相當接近原先的期望值 4 倍，可以說明系統對於檔案分配、結果儲存的額外負擔小於整體執行的 0.1 倍。

圖表 27 為此實驗每 50 分鐘處理個數的直方圖，根據此圖可說明此系統的效能並不會因為時間或處理個數的增加而減少，維持穩定的產出，足以說明系統的可行性。而由圖表 28 和圖表 27，也可以說明系統的效能會隨著分析器的增加而上升，具有相當高的可擴充性。



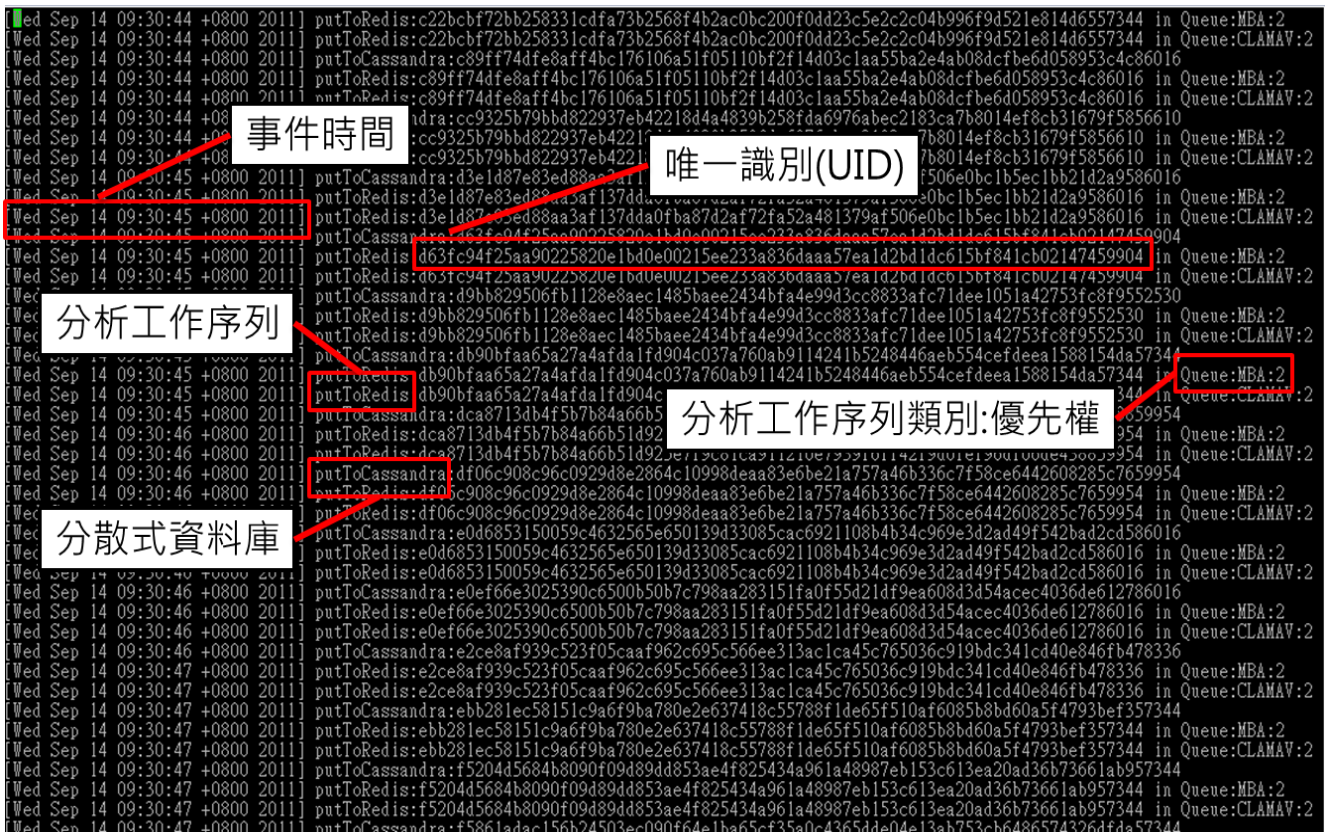
圖表 23: 每 50 分鐘分析個數直方圖。



圖表 24: 時間-分析個數關係圖。

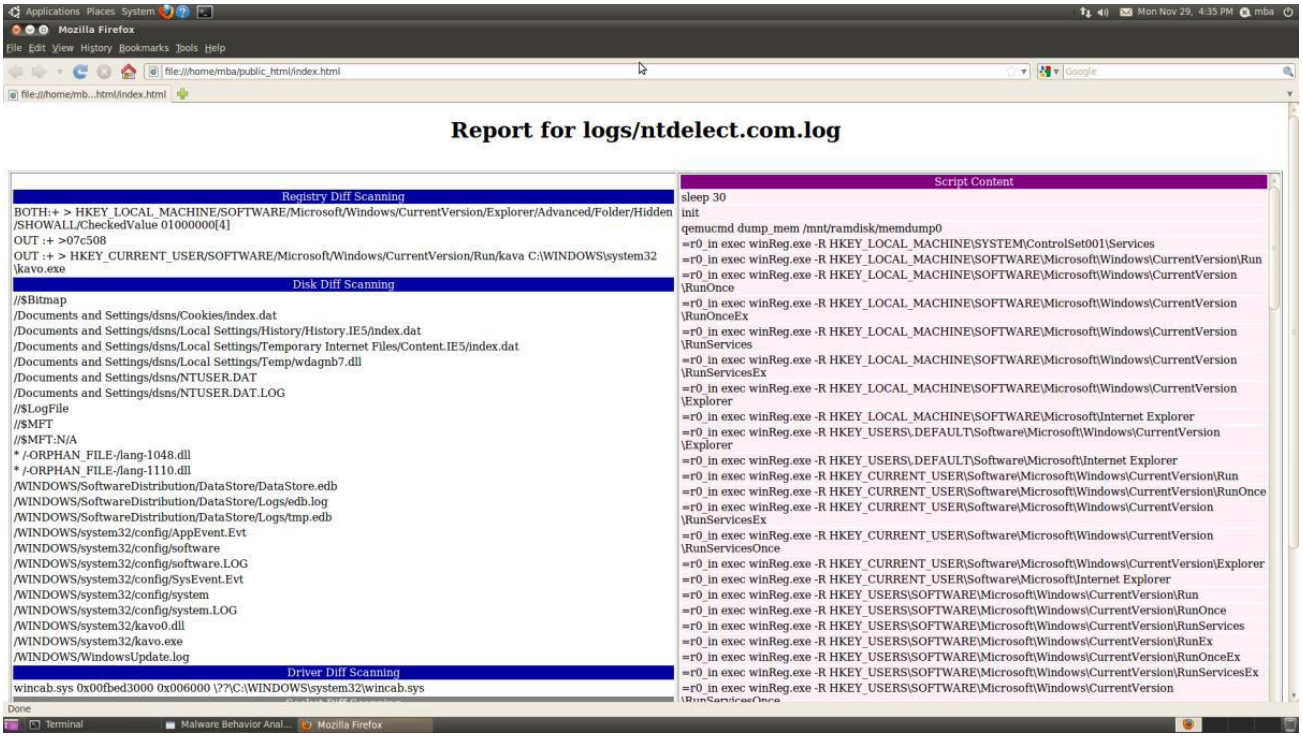
■ 運行截圖

圖表 25 分析過程之紀錄檔案截圖



圖表 26 放分析工做至惡意程式分析雲端平台

圖表 27 網頁爬取子系統之紀錄截圖



圖表 28 實際的惡意樣本分析報告

肆、 技術方案優越性

雲端惡意程式分析平台包含下列四項關鍵技術：１）雲端節點機器管理、２）分析工作分配與排程、３）可擴充式掛載介面、４）動態汙染源技術，以下將條列式介紹。

雲端節點機器管理

不同於市面上廣為人知的雲端虛擬化節點管理，此子系統需管理實體電腦。市面上的雲端服務大多以租用運算環境來營利，此種服務的環境被美國國家標準與技術單位(National Institute of Standards and Technology)稱之為「公開雲 (public cloud)」。這種公開讓租用者使用的管理介面可以利用虛擬化的技術來達成。而本子系統的開發環境並非隨意讓使用者隨意存取內部節點之運算能力，這種基於利用雲端來增強運算能力所形成的雲端環境可被稱作「內部雲(private cloud)」。因為環境的不同，所以本平台的節點並非利用現有虛擬化技術來做管理，而是整合網路監控、系統監控來實作真實機器管理工作。

由於此平台包含許多實體的電腦，為了維護方便，此平台必須具備有容錯機制以及自我修復機制。在容錯機制部分，此平台可能遭受到許多種類的錯誤，例如：網路斷線、停電、系統錯誤等。如果沒有容錯機制的話，可能會因為單一節點的錯誤而造成整個系統無法繼續運行。在系統設計上，需要針對有可能切割的部分做容錯機制，來增加此系統的穩定性。在自我修復機制方面，是為了能夠讓此系統從簡單的錯誤狀態，不藉著人工修復方式來使系統回復到可運行的階段。如此一來就可以減少人力成本，也可以縮短系統修復的時間開銷。

藉著取得每個運算節點上的機器狀態，可以動態調整每台電腦上的工作以最大化此分析平台的產能。因為每一個節點的運算能力可能不同，如果只是讓分析工作平均的分散到各分析節點上，則可能會有機器閒置的情形。雲端節點機器管理需要綜觀各節點的使用狀態來分配工作。

分析工作分配與排程

此平台包含各種功能性節點，例如：蒐集檔案節點、惡意程式分析器節點、資料庫節點等，從建立分析工作到分析結果出現，中間透過了許多的網路溝通。而這些工作分配是否能有效率地傳輸，是本平台提高分析產能的重要技術之一。為了節省空間使用，盡量避免分析相同的檔案或是多餘的檔案複製等。同時，減少硬碟讀寫次數和系統輸入輸出能夠大量的增加檔案分析的速度。為了達到以上的目的，本系統將會把部分的資料保存在記憶體中，以減少讀寫的時間開銷。

分析工作的排程能夠縮短反應時間。每個被產生的分析工作都有它的優先權，以實行不同的反應時間。例如，背景執行的網路擷取的檔案可以隨時被暫停，空出資源來回應使用者查詢的檔案檢測。優先權的高低差異讓本系統可以有彈性的去管理分析工作，動態地調動運算資源給較緊急的分析工作。

可擴充式掛載介面

程式分析可以用許多不同面向的分析技術來判別該樣本是否為惡意。又隨著攻擊手法的日新月異，要持續地維護這些分析工具才能符合時下的需求。一旦有新的分析技術的產生，系統開發人員必須花費時間在機器環境的更新以及架設，十分耗費人力。本子系統欲設計一個可擴充式掛載介面，可以輕易地安裝新形態的分析工具，移除較為老舊的版本。為了達到此目的，我們需要制定一套惡意程式分析的框架，來幫助系統知道各分析工具的運行方式、初始化和資源需求等，利用一個通用的方式來描述分析工具的作業流程。其部分包含如何運行的指令、傳入的參數、套件的安裝、環境需求和結束時資源歸還等。設計好這個可擴充式掛載介面，往後就可以供相關的學術研究使用，掛載自行開發的分析工具，以驗證分析方法的正確性。

動態汙染源技術

本分析系統的關鍵技術在於汙染源資訊流動追蹤技術上。在一般的動態分析研究中，所分析的目標大多為單一的、運行於使用者層次的程式，因此分析系統所需提供的模擬環境較簡易。但全系統層次行為分析系統卻是追蹤整體作業系統運行該程式的資訊流動。動態汙染分析是由以往的資訊流追蹤(Information Flow Tracking)技術演變而來。在以往的資訊流追蹤研究中，是將作業系統中如使用者、檔案、程序、網路連線…等視為物件，並記錄這些物件間的資訊流動情形，以偵測是否有違反資安規則的行為發生。例如當某位使用者 U 執行某程序 P 對某檔案 F 寫入資料時，便發生了由 U 到 F 的資訊流動，如 F 不為 U 所有，則為一異常行為。此外，從上面的例子可了解，資訊流動的追蹤包含的三個要素為：物件、操作行為與資安規則；物件乃資訊流動的起點、終點或中繼站，而操作行為是造成資訊流動的原因，資安規則是正常與異常資訊流動的分界。在上例中包含了三個物件(U、P 和 F)，以及兩個引起資訊流動的操作行為(執行與寫入)，而「不可寫入他人檔案」為資安規則。事實上，絕大多數違反資安的行為皆可以異常的資訊流動加以解釋。

雖然資訊流追蹤是十分有用的分析方法，但追蹤的物件細緻度決定了分析的準確度。當以越高層次的物件如檔案、使用者做為分析對象時，將失去許多於同一物件中資訊流動的訊息。然而分析越低層次的物件如CPU暫存器或記憶體位元組時，所需記錄的訊息也就越多，效能也將大打折扣。因此在以往電腦效能緩慢的時代，並無法進行這麼細膩的分析，現今的電腦效能已足以進行低層次的資訊流追蹤工作，亦即本計畫著重的動態汙染分析技術。

動態汙染分析的物件為 CPU、記憶體以及硬碟中的每個位元組，換言之系統中每個資料位元組皆被視為一單獨的分析單位，因此已達到相當精細的分析準確度(最準確的分析必須以位元為分析單位，但空間負擔過高，如後述)。而位元組 A 至位元組 B 的資訊流動稱為 A 對 B 的「污染」，造成污染的操作行為則為 CPU 執行的每一行指令。資安規則是透過「污染源」與「污染目標」來定義，一旦發生從污染源至污染目標的資訊流動，即代表異常行為的發生。以上即為動態汙染分析的基本原理

由於 X86 CPU 為 CISC 架構，故指令集十分複雜共包含數百道不同功能的指令。以往針對單一應用程式的動態污染分析，由於應用程式本身只能使用 Ring 3 層級的指令，因此所需模擬其中的數十道指令。但本研究計劃為求分析的完整與可應用性，必須對數百道指令加上如上表中的追蹤動作，工程相當繁複。

伍、 結論與展望

雲端運算提供了新型態的運算架構以及龐大資料的儲存技術。藉著架設多台節點所形成的惡意程式分析雲端平台，可以有效地利用其運算資源，以提供大量的分析產能。由於國內學術界內尚未有相關的惡意程式分析平台，故本研究將著力於研究雲端分析平台之系統架構，以及實際地開發出一套可結合不同分析模組的雲端系統。開發完成的雲端系統，可以提供龐大的運算資源，供惡意程式研究之相關單位使用，或是提供實際的惡意樣本分析服務。

此平台有幾項重點功能，首先為網頁爬取子系統，該子系統將會持續性地蒐集網路上可連結之資料，並且下載其可執行檔案至分析平台內進行分析。不同以往被動式的誘捕系統，此方式主動地去尋找可能受感染的網站或檔案，以搜尋最新的惡意程式樣本。目前有架設兩台網頁爬取節點，單日可瀏覽約二十萬筆的網路連結。藉著 Map-Reduce 的運算技術，可以過濾重複之檔案連結，以減少重複的檔案爬取。其次，本平台結合了最新雲端儲存的分散式資料庫系統，用來大量地儲存、備份分析過的可執行檔案。由於分散式資料庫系統的查詢時間非常迅速，所以可以用來做大量資料的處理、查詢。接下來，為了整合不同面向的分析工具，本平台提供了一套有彈性的整合介面，讓分析人員能夠快速地整合不同種類的分析工具。而該分析平台能夠做良好的工作分配，讓平台的產能為最大化。

目前此平台已整合本實驗室所開發之全系統層次分析系統。此分析技術因動態地運行惡意程式樣本，所以需要耗費大量的運算資源和記憶體。也因此，上述之分析模組不適合安裝在一般的個人電腦之中。藉著抽象化分析流程來達到惡意程式的擷取、儲存和分析，可以讓此分析平台成為一個強大私有雲架構的分析器。透過本計畫所開發之惡意程式分析雲端平台，可以將要檢驗的可疑樣本，全數放到雲端分析伺服器上。再由系統自動分配其分析工作，以快速地達成分析的目的。除此之外，惡意程式分析人員，也可自行開發其特有的分析模組，透過分析模組掛載介面，安裝於此平台上，以驗證其分析效果。

最後，我們設計一套實驗來驗證我們的系統。實驗結果顯示此系統對分析器僅增加相當輕量的負擔，可以發揮運算節點本身的效能。並可隨著節點增加提高整體產量，具有良好的可擴充性。並且工作分析處理時間不因整體處理個數或時間而增加，因此本系統也具備相當的實用性。藉此研究計畫，我們可以得知搭配雲端技術所帶來新的特性能夠強化以往傳統分析的不足。

隨著今年頒發了「個資法」，可突顯國內政府產業逐漸重視資訊安全。而雲端的技術興起，國內學者業界也趨之若鶩。本計畫為了加強現有惡意程式分析技術，使得資訊安全與雲端技術產生新的火花，來補強以往所無法達成的目標：兼顧效率以及準確性。藉由著本計畫所衍伸技術的產學合作、專利申請、論文發表的成果來看，更可證明了本計畫不論在研究深度、研究價值或實作門檻，將會受到國內資訊安全學界的重視。這樣新穎且具有創新思考的研究方向，確實值得投入大量人力以完成更豐碩的成果。也因此，本計畫也培育了重要的資安人才，不僅在主持人謝續平教授或是實驗室同學們，都有在國內資安領域有著令人注目的成績，這些培育得優秀人才能更加地鞏固國內資訊安全領域的實力。

陸、參考文獻

[1] R. Perdisci, A. Lanzi, and W. Lee, "McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables," in *Computer Security Applications Conference, Annual*, Los Alamitos, CA, USA, 2008, vol. 0, pp. 301-310.

[2] P. Baecher, M. Koetter, M. Dornseif, and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," *IN PROCEEDINGS OF THE 9 TH INTERNATIONAL SYMPOSIUM ON RECENT ADVANCES IN INTRUSION DETECTION (RAID*, p. 165--184, 2006.

[3] Y. Ye, T. Li, Y. Chen, and Q. Jiang, "Automatic malware categorization using cluster ensemble," *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 95–104, 2010.

[4] S. Muhlbach and A. Koch, "A Dynamically Reconfigured Network Platform for High-Speed Malware Collection," *Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs*, p. 79–84, 2010.

[5] C. Leita and M. Dacier, "SGNET: A Worldwide Deployable Framework to Support the Analysis of Malware Threat Models," *Proceedings of the 2008 Seventh European Dependable Computing Conference*, p. 99–109, 2008.

[6] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," *Proceedings of the 10th international conference on Recent advances in intrusion detection*, p. 178–197, 2007.

[7] G. Wicherski, "peHash: a novel approach to fast malware clustering," *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, p. 1–1, 2009.

[8] Y. Ye, T. Li, Q. Jiang, Z. Han, and L. Wan, "Intelligent file scoring system for malware detection from the gray list," *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 1385–1394, 2009.

[9] V. Mulukutla, "Wolfsting: Extending Online Dynamic Malware Analysis Systems by Engaging Malware.," 2010.

[10] M. F. Zolkipli and A. Jantan, "Malware Behavior Analysis: Learning and Understanding Current Malware Threats," in *2010 Second International Conference on Network Applications, Protocols and Services*, 2010, p. 218–221.

[11] X. Jiang, Z. Hao, and Y. Wang, "A Malware Sample Capturing and Tracking System," in *2010 Second WRI World Congress on Software Engineering*, 2010, p. 69–72.

[12] D. Cavalca and E. Goldoni, "An Open Architecture for Distributed Malware Collection and Analysis," *Open Source Software for Digital Forensics*, p. 101–116, 2010.

[13] Z. Tzermias, G. Sykiotakis, M. Polychronakis, and E. P. Markatos, "Combining Static and Dynamic Analysis for the Detection of Malicious Documents," 2011.

[14] M. Apel, J. Biskup, U. Flegel, and M. Meier, "Early Warning System on a National Level–Project AMSEL," in *Proceedings of the First Workshop on Early Warning and Network Intelligence (EWNI), Hamburg, Germany*, 2010.

[15] B. J. Nabholz, "Design of an Automated Malware Analysis System," *College of Technology Directed Projects*, p. 4, 2010.

[16] Y. M. Wang, D. Beck, X. Jiang, and R. Roussev, "Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities," in *IN NDSS*, 2006.

[17] Anubis: Analyzing Unknown Binaries, http://anubis.iseclab.org/

[18] James P. Anderson, Computer Security Technology Planning Study, Volumn II. 1972

[19] Frédéric Perriot, Peter Ferrie and Péter Ször, Virus Bulletin. Symantec Security Response, USA. May 2002

[20] Orr, "The Molecular Virology of exotan32 Metamorphism Illustrated." [Online document] July. 2007,

[21] Available at http://www.antilife.org/files/Lexo32.pdf

[22] Greg Hoglund and Jamie Butler, Rootkits: Subverting the Windows Kernel, Addison-Wesley Professional, August 1, 2005.

[23] David Brumley, Pongsin Poosankam, Dawn Song, Jiang Zheng. "Automatic Patch-Based Exploit Generation is Possible: Techniques and Implications". 2008 IEEE Symposium on Security and Privacy, April, 2008;

[24] Holz, T., Steiner, M., Dahl, F., Biersack, E., and Freiling, F. “Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm.” Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (San Francisco, California, April 15 - 15, 2008). F. Monrose, Ed. USENIX Association, Berkeley, CA, 1-9.

[25] Yin, H., Song, D., Egele, M., Kruegel, C., and Kirda, E. “Panorama: capturing system-wide information flow for malware detection and analysis.” In Proceedings of the 14th ACM Conference on Computer and Communications Security (Alexandria, Virginia, USA, October 28 - 31, 2007). CCS '07. ACM, New York, NY, 116-127.

[26] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha. “Towards automatic generation of vulnerability-based signatures.” In Proceedings of the IEEE Symposium on Security and Privacy, 2006.

[27] Costa, M., Crowcroft, J., Castro, M., Rowstron, A., Zhou, L., Zhang, L., and Barham, P. “Vigilante: end-to-end containment of internet worms.” In Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (Brighton, United Kingdom, October 23 - 26, 2005). SOSP '05. ACM, New York, NY, 133-147.

[28] Kreibich, C. and Crowcroft, J. “Honeycomb: creating intrusion detection signatures using honeypots.” SIGCOMM Comput. Commun. Rev. 34, 1 (Jan. 2004), 51-56. DOI= http://doi.acm.org/10.1145/972374.972384

[29] Lance Spitzner, "Honeypots: Catching the Insider Threat," Computer Security Applications Conference, Annual, vol. 0, no. 0, pp. 170, 19th Annual Computer Security Applications Conference (ACSAC '03), 2003.

[30] Ruwase, O., Gibbons, P. B., Mowry, T. C., Ramachandran, V., Chen, S., Kozuch, M., and Ryan, M. “Parallelizing dynamic information flow tracking.” In Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures (Munich, Germany, June 14 - 16, 2008). SPAA '08. ACM, New York, NY, 35-45.

[31] Edmund B. Nightingale, Daniel Peek, Peter M. Chen, Jason Flinn. “Parallelizing security checks on commodity hardware.” Architectural Support for Programming Languages and Operating Systems (ASPLOS 2008), March 2008.

[32] Guru Venkataramani, Ioannis Doudalis, Yan Solihin and Milos Prvulovic. “FlexiTaint: A Programmable Accelerator for Dynamic Taint Propagation.” Proc. of the 14th International Symposium on High Performance Computer Architecture (HPCA), Feb 2008.

[33] Nethercote, N. and Seward, J. “Valgrind: a framework for heavyweight dynamic binary instrumentation.” SIGPLAN Not. 42, 6 (Jun. 2007), 89-100. DOI= http://doi.acm.org/10.1145/1273442.1250746

[34] Lawton, K. P. “Bochs: A Portable PC Emulator for Unix/X.” Linux J. 1996, 29es (Sep. 1996), 7.

[35] Bellard, F. “QEMU, a fast and portable dynamic translator.” In Proceedings of the Annual Conference on USENIX Annual Technical Conference (Anaheim, CA, April 10 - 15, 2005). USENIX Annual Technical Conference. USENIX Association, Berkeley, CA, 41-41.

[36] VICE, J. Butler and G. Hoglund. “VICE–catch the hookers!” In Black Hat USA, July 2004.

[37] J. Rutkowska. “System virginity verifier: Defining the roadmap for malware detection on windows systems,” in Hack In The Box Security Conference, September 2005.

[38] Heng Yin, Zhenkai Liang, and Dawn Song. “HookFinder: Identifying and Understanding Malware Hooking Behaviors.” in 15th Annual Network and Distributed System Security Symposium (NDSS'08), February 2008.

[39] Min Gyung Kang, Pongsin Poosankam, and Heng Yin. “Renovo: A Hidden Code Extractor for Packed Executables,” in 5th ACM Workshop on Recurring Malcode (WORM), October 2007.

[40] James Newsome and Dawn Song. “Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploit Attacks on Commodity Software,” in Network and Distributed Systems Security Symposium, Feb 2005.

[41] G. Edward Suh, Jaewook Lee and Srinivas Devadas. “Secure Program Execution via Dynamic Information Flow Tracking,” in CSAIL Technical Reports, July 2003

[42] Michael Dalton, Hari Kannan, and Christos Kozyrakis. “Real-World Buffer Overflow Protection for Userspace & Kernelspace,” in 17th USENIX Security Symposium, 2008

[43] Michael Dalton, Hari Kannan, and Christos Kozyrakis. “Raksha: a flexible information flow architecture for software security,” in Technical Record of the 19th Hot Chips Symposium, Palo Alto, CA, August 2007

國科會補助計畫衍生研發成果推廣資料表

日期:2012/10/31

國科會補助計畫	計畫名稱: 子計畫二：基於機器碼之Windows惡意程式行為分析雲端平台(2/2)		
	計畫主持人: 謝續平		
	計畫編號: 100-2218-E-009-004- 學門領域: 資訊		
研發成果名稱	(中文) 利用多重執行路徑比較尋找程式分歧點以偵測虛擬機器感知之惡意軟體		
	(英文) Detecting VM-Aware Malware Using Divergent Multi-Execution Traces		
成果歸屬機構	國立交通大學	發明人(創作人)	謝續平, 許家維, 李秉翰
技術說明	(中文) 一種具虛擬機器感知能力程式之偵測方法包含以下步驟：藉由至少一實體機器，提供數個虛擬機器。其中，虛擬機器之至少其中之二為異質虛擬機器。接收具感知虛擬機器功能之一樣品程式。使每一虛擬機器分別執行樣品程式。分別取得每一虛擬機器執行樣品程式之至少一執行追蹤紀錄。比對執行追蹤紀錄，以取得執行追蹤紀錄間所存在之至少一感知能力偵測點。接收一待驗證程式。判斷待驗證程式是否存在感知能力偵測點。其中，在待驗證程式存在感知能力偵測點時，判定待驗證程式具虛擬機器感知功能。		
	(英文) A method for detecting a program with a virtual machine (VM) aware ability includes the following steps: several VMs are provided through at least one physical machine. Wherein, at least two of the VMs are heterogeneous. A sampling program with a VM aware ability is received. Each of the VMs is driven to execute the sampling program respectively. At least one execution trace record, which records information about each VM executing the sampling program. The execution trace records are compared to obtain at least one check point between the execution trace records. A program to be checked is received. Determine if the program to be checked includes the check point. Wherein, if the program to be checked includes the check point, the program to be checked is determined as the one with the VM aware ability.		
產業別	其他專業、科學及技術服務業		
技術/產品應用範圍	可能應用之產業(applicable industries)： Anti-Virus, 資訊安全, 網路安全, 雲端服務產業, 電信服務提供產業 可能應用之產品(applicable products)： Anti-Virus, Malware Analysis, 病毒行為分析, 病毒偵測		
技術移轉可行性及預期效益	可以將技術移轉，協助單位判斷具虛擬機器感知能力之惡意程式，使分析惡意程式能力更為準確。		

註：本項研發成果若尚未申請專利，請勿揭露可申請專利之主要內容。

100 年度專題研究計畫研究成果彙整表

計畫主持人：謝續平	計畫編號：100-2218-E-009-004-
計畫名稱：前瞻性雲端安全儲存、防護、行為分析與觀測平台--子計畫二：基於機器碼之 Windows 惡意程式行為分析雲端平台(2/2)	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	3	3	60%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	2	2	50%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	2	2	100%	人次	
		博士生	2	2	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	4	4	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	2	2	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	3	3	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	學生獲獎 1. 2010 Hacks in Taiwan Conference 台灣駭客年會 Wargame 競賽冠軍 2. 2011 Hacks in Taiwan Conference 台灣駭客年會 Wargame 競賽冠軍 3. 2012 Hacks in Taiwan Conference 台灣駭客年會 Wargame 競賽殿軍 4. 2011 資策會資安技能金盾獎亞軍 主持人社會服務 1. 總統府國家安全會議顧問 2. 國家資通安全會報技服中心惡意程式交流聯盟主席

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估 ■達成目標 □未達成目標（請說明，以 100 字為限） □實驗失敗 □因故實驗中斷 □其他原因 說明：
2. 研究成果在學術期刊發表或申請專利等情形： 論文：□已發表 ■未發表之文稿 □撰寫中 □無 專利：□已獲得 ■申請中 □無 技轉：□已技轉 ■洽談中 □無 其他：(以 100 字為限)
3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)