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、偵測、行為分析與觀測--總計畫(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-009-003-
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：曾文貴
共 同 主 持 人：謝續平、黃育綸、吳育松
計畫參與人員：碩士級-專任助理人員：張智凱
碩士班研究生-兼任助理人員：賴托登
碩士班研究生-兼任助理人員：陶嘉仁
碩士班研究生-兼任助理人員：楊其仁

公 開 資 訊 ：本計畫可公開查詢

中 華 民 國 101 年 10 月 30 日

中文摘要：在網路與行動裝置的催化下，雲端服務已是IT產業的重點技術，因此雲端的安全問題就顯得格外重要。本計畫延續之前與美國柏克萊大學研究團隊在虛擬機器資安跨國合作研究之經驗與成果為基礎，提出創新的方法並建構前瞻性雲端安全儲存、防護、行為分析與觀測平台。此平台包含四個子計畫：曾文貴教授所主持的子計畫「支援多樣功能之雲端資料安全儲存」、謝續平教授所主持的子計畫「基於機器碼之Windows惡意程式行為分析雲端平台」、黃育綸教授所主持的子計畫「設計與實作基於雲端技術之安全實驗觀測網路」、與吳育松教授所主持的子計畫「基於Xen Hypervisor之即時雲端環境入侵偵測與反制」。本平台不但能提供雲端儲存的安全性、正確性及其他功能性，亦可透過即時入侵偵測及反制系統確保整個雲端平台不會受到一般的網路攻擊，並能利用雲端的運算能力動態分析經過變形、隱匿、加殼等處理的複雜攻擊行為。本計劃更提供一個具有仿真性與即時性的實驗觀測網路平台，不但可直接取得接近硬體層級的網路流量資訊、模擬受到攻擊的網路狀態、更能讓需要改變的網路拓樸即時生效。

本系統整體運作方式簡述如下：來自於Internet的連線將會先經過即時雲端環境入侵偵測與反制子系統，由此子系統負責針對已知的攻擊類型進行防阻，以提供一個安全的雲端平台環境。而建構在此雲端平台之上的雲端資料安全儲存子系統可保證儲存資料的正確性與私密性，並可提供加密後資料的比對搜尋與再授權，並藉此提供各種應用功能。針對已儲存在此雲端儲存系統中的各類型檔案，本系統亦提供惡意程式行為分析子系統。有別於傳統的特徵式分析，此子系統著重在惡意程式動態執行階段的行為模式分析，以找出可能惡意行為之特徵，並回饋至第一線的即時雲端環境入侵偵測與反制子系統。較為複雜的攻擊行為也可利用本計畫所開發的實驗觀測網路平台進行實驗與觀測，不但有助於針對複雜行為提供進一步的分析，其高度隔離的環境亦可確保具危險性的實驗不會干擾到其他系統的正常運作。

綜合來看，本計畫建構一個包含多面向的雲端安全服務平台，針對雲端資料安全儲存、即時雲端環境入侵偵測與反制、惡意程式行為動態分析、仿真性網路實驗觀測平台等領域提出理論與實務創新的構想與設計，且相關研究成果已與多家知名廠商與政府機構簽訂技術合作與產學計畫，足見其實用價值。

中文關鍵詞：雲端計算、雲端儲存、入侵偵測、入侵預防、動態汙染分

析、惡意行為分析、雲端虛擬化技術、實驗觀測網路

英文摘要： With the growth of Internet and mobile devices, cloud service had become a key point of networking technologies. The trend makes security issues more and more important. This project proposes an integrated solution including four sub-projects to research and design a secure cloud platform. Prof. Wen-Guey Tzeng leads the sub-project to design a secure muti-functional cloud storage system. Prof. Shiuhpyng Shieh leads the sub-project to develop a dynamic analyzing cloud against complex malwares. Prof. Yu-Lun Huang leads the sub-project to develop an observation cloud for security experiments. Prof. Yu-Sung Wu leads the sub-project to integrate IDS/IPS into Xen Hypervisor. The integrated solution proposed by this project not only provides correctness, privacy, and other security functions for cloud storage. Its IPS also provides a first line of defense against incoming attacks. Moreover, it has the ability to dynamically analyze complex malicious behaviors using the computing power of a private cloud and feed the signature of the attacks back to the IPS. Furthermore, the complex attacks can be securely experimented in the observation cloud. In conclusion, this project constructs a cloud platform with four main sub-systems: a set of efficient solutions to the security problems in cloud storage systems, a malware analysis system based on cloud computing, an IDS/IPS-integrated Xen Hypervisor, and an observation cloud for security experiments. New theories were proposed and the related systems were implemented. This project also directly contributed to various cooperative plans between the industry and the academic. The cooperative plans also show that this project is valuable and feasible.

英文關鍵詞： cloud computing, cloud storage, intrusion detection, intrusion prevention, malware analysis, taint analysis, virtualization

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

計畫類別：□個別型計畫　■整合型計畫
計畫編號：NSC 100-2218-E-009-003
執行期間： 100 年 08 月 01 日 至 101 年 07 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：曾文貴
共同主持人：謝續平、黃育綸、吳育松

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

在網路與行動裝置的催化下，雲端服務已是 IT 產業的重點技術，因此雲端的安全問題就顯得格外重要。本計畫延續之前與美國柏克萊大學研究團隊在虛擬機器資安跨國合作研究之經驗與成果為基礎，提出創新的方法並建構前瞻性雲端安全儲存、防護、行為分析與觀測平台。此平台包含四個子計畫：曾文貴教授所主持的子計畫「支援多樣功能之雲端資料安全儲存」、謝續平教授所主持的子計畫「基於機器碼之 Windows 惡意程式行為分析雲端平台」、黃育綸教授所主持的子計畫「設計與實作基於雲端技術之安全實驗觀測網路」、與吳育松教授所主持的子計畫「基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制」。本平台不但能提供雲端儲存的安全性、正確性及其他功能性，亦可透過即時入侵偵測及反制系統確保整個雲端平台不會受到一般的網路攻擊，並能利用雲端的運算能力動態分析經過變形、隱匿、加殼等處理的複雜攻擊行為。本計劃更提供一個具有仿真性與即時性的實驗觀測網路平台，不但可直接取得接近硬體層級的網路流量資訊、模擬受到攻擊的網路狀態、更能讓需要改變的網路拓樸即時生效。

本系統整體運作方式簡述如下：來自於 Internet 的連線將會先經過即時雲端環境入侵偵測與反制子系統，由此子系統負責針對已知的攻擊類型進行防阻，以提供一個安全的雲端平台環境。而建構在此雲端平台之上的雲端資料安全儲存子系統可保證儲存資料的正確性與私密性，並可提供加密後資料的比對搜尋與再授權，並藉此提供各種應用功能。針對已儲存在此雲端儲存系統中的各類型檔案，本系統亦提供惡意程式行為分析子系統。有別於傳統的特徵式分析，此子系統著重在惡意程式動態執行階段的行為模式分析，以找出可能惡意行為之特徵，並回饋至第一線的即時雲端環境入侵偵測與反制子系統。較為複雜的攻擊行為也可利用本計畫所開發的實驗觀測網路平台進行實驗與觀測，不但有助於針對複雜行為提供進一步的分析，其高度隔離的環境亦可確保具危險性的實驗不會干擾到其他系統的正常運作。

綜合來看，本計畫建構一個包含多面向的雲端安全服務平台，針對雲端資料安全儲存、即時雲端環境入侵偵測與反制、惡意程式行為動態分析、仿真性網路實驗觀測平台等領域提出理論與實務創新的構想與設計，且相關研究成果已與多家知名廠商與政府機構簽訂技術合作與產學計畫，足見其實用價值。

關鍵詞：雲端計算、雲端儲存、入侵偵測、入侵預防、動態汙染分析、惡意行為分析、雲端虛擬化技術、實驗觀測網路

Abstract

With the growth of Internet and mobile devices, cloud service had become a key point of networking technologies. The trend makes security issues more and more important. This project proposes an integrated solution including four sub-projects to research and design a secure cloud platform. Prof. Wen-Guey Tzeng leads the sub-project to design a secure muti-functional cloud storage system. Prof. Shiuhpyng Shieh leads the sub-project to develop a dynamic analyzing cloud against complex malwares. Prof. Yu-Lun Huang leads the sub-project to develop an observation cloud for security experiments. Prof. Yu-Sung Wu leads the sub-project to integrate IDS/IPS into Xen Hypervisor. The integrated solution proposed by this project not only provides correctness, privacy, and other security functions for cloud storage. Its IPS also provides a first line of defense against incoming attacks. Moreover, it has the ability to dynamically analyze complex malicious behaviors using the computing power of a private cloud and feed the signature of the attacks back to the IPS. Furthermore, the complex attacks can be securely experimented in the observation cloud.

In conclusion, this project constructs a cloud platform with four main sub-systems: a set of efficient solutions to the security problems in cloud storage systems, a malware analysis system based on cloud computing, an IDS/IPS-integrated Xen Hypervisor, and an observation cloud for security experiments. New theories were proposed and the related systems were implemented. This project also directly contributed to various cooperative plans between the industry and the academic. The cooperative plans also show that this project is valuable and feasible.

Keywords: cloud computing, cloud storage, intrusion detection, intrusion prevention, malware analysis, taint analysis, virtualization,

目 錄

圖表目錄

壹、 計畫內容及目的

一、 前言及背景說明

在技術發展史上來看，分散式系統已經有初期的雲端概念，然而當時並沒有普及的網路架構與隨身可上網的裝置作為基礎，現今完善的網路環境則啟動了雲端技術的蓬勃發展。整個雲端的架構可以說是將早期的大型主機與終端機架構中主機到終端機的連線改為網路媒介，另外將單一主機改為大型伺服器群組。雲端議題之所以倍受關注，它表彰的精減成本精神是最切合企業需要的部份，更呼應了當前全球節能減碳的大趨勢。

雲端帶來了許多好處，技術瓶頸慢慢被突破，雲端服務已經開始充斥在一般民眾的日常生活之中。但是當資料大量集中到雲端裡面的時候，就衍生了許多安全議題。有別於以往個人電腦的安全問題，在雲端裡面若發生安全性的問題，後果可能是影響到所有在雲端活動的使用者。因此，安全性要求是雲端系統必備的條件之一，而除了保障使用者儲存的資料不會在未經許可下被窺視和竄改外，若系統能提供多樣性的功能（如錯誤修正、內容分享、存取控制等）將使得雲端基礎建設功能性更加完備。

除此之外，由於現今的網路環境充斥著各種攻擊行為，傳統的做法是由各主機或服務自行建構入侵偵測及防護系統，這樣的做法在雲端環境並不可行，其理由是雲端平台上的服務與虛擬主機所建構的各式入侵偵測及防護系統不但功能上互相重複，且也會浪費大量的資源並明顯的拖累整體的系統效能。相較之下，將入侵偵測與防護系統建構在底層的雲端平台是一個較有效率的設計，由於底層的雲端架構已將網路中已知的一般攻擊隔絕，其上層的各系統可以專注於提供各自的服務，不但可以省下大量的投資，亦可解決運算資源被浪費在重複架設的入侵偵測防護系統之上。

不同於已知的一般攻擊，傳統透過特徵偵測或靜態分析的技術對垃圾郵件與數量更為龐大且變化更為快速的病毒成效不彰，新型的攻擊手法亦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。

目前分析技術可分成靜態分析（static analysis）與動態分析（dynamic analysis）兩大類，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該檔案的內容，或是能夠判斷該檔案是否有可疑的加殼或加密行為。而動態分析而是直接模擬運行時的狀態，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。由於動態分析需要耗費大量的運算能力，故如何利用雲端的運算彈性來進行動態分析亦為一個相當有價值得研究課題。

此外，雲端計算的基礎技術強調如何利用虛擬化及自動化技術，以實現運算資源最佳化與資源共享等概念，例如利用虛擬機器 Xen 或 KVM，依使用者需求配置運算資源等。然而，目前雲端計算所採用之基礎技術卻尚不足以支援即時性（real-time）與仿真性（fidelity），也很難滿足某些特定應用或服務的需求，在網路測試服務上面更是無法提供所需的種種特性。

綜合以上所述，雲端技術為目前資訊科技的重要趨勢，故如何提供一個安全的雲端環境是目前最重要的研究課題之一。本計畫的研究主題可分為四個部分，分別為：支援多樣功能之雲端資料安全儲存、全系統層次動態惡意程式行為分析雲端平台、基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制、與基於雲端技術之安全實驗觀測網路。各主題之背景說明如下：

- **支援多樣功能之雲端資料安全儲存**

雲端運算以及行動資訊服務是近年來非常熱門的商業模式，同時也是未來的資訊服務發展的趨勢。隨著網際網路的快速發展以及行動裝置的進步，現今已有許多相關的雲端運算服務已出現在人們的日常生活當中，如 Gmail、Dropbox 和 Facebook 等。雲端運算是一個根據使用者要求提供服務的模型，使用者運算和儲存的資料都可以不用放在本地端。這樣的模型可以大幅降低使用者裝置所需的運算能力及儲存空間，使用者可透過高速的電信與資訊網路系統並利用個人終端設備在任何時間任何地點取得資訊服務業者在雲端上所提供的資源和服務。當資料大量集中到雲端裡面的時候，就衍生了許多安全議題。有別於以往個人電腦的安全問題，在雲端裡面若發生安全性的問題，後果可能是影響到所有在雲端活動的使用者。因此，安全性要求是雲端系統必備的條件之一，而除了保障使用者儲存的資料不會在未經許可下被窺視和竄改外，若系統能提供多樣性的功能（如錯誤修正、內容分享、存取控制等）將使得雲端基礎建設功能性更加完備。

此研究主題為支援多樣功能之雲端資料安全儲存。在考量使用者資料隱私性以及雲端儲存資料安全性下，此部份的研究主要可分為以下三項：

(1) 保護隱私的雲端入侵偵測

雲端運算改變了安全軟體服務的模式。對於網路上層出不窮的攻擊，傳統的安全軟體都是執行在個人的電腦上，使用者可以在不用傳送任何資料到網路上的情形下進行安全檢測以防範網路上各種不同的攻擊。而在防範攻擊的方法中，最有效也最多人使用的方式就是入侵偵測系統(Intrusion Detection System)。近幾年來已有一些公司開始提供雲端入侵偵測的服務給使用者使用。這些服務大都是將病毒碼放在雲上，使用者只需先行掃描自己的系統，然後再掃描出來的資訊傳送至雲端進行分析、比對並得知結果。這樣的方法可降低使用者個人行動運算裝置的運算和儲存量，同時也省去了使用者需要時常自行更新病毒碼的麻煩。然而，現有的雲端入侵偵測的服務都沒有考慮到使用者傳送到雲端的資訊可能會涉及使用者隱私，這些資料除了可能會被雲端服務端濫用或遭到洩漏外，在傳送的過程中也有可能會造成資料的外洩。

(2) 非集中式雲端儲存系統中系統修復機制

無論是早期的網路儲存系統或新興的雲端儲存系統，都要面對"提供不間斷的服務"的挑戰。垂直層面上來看，軟體有可能出錯，網路有可能中斷，硬體也有可能會無預警毀損。於是容錯技術被應用到網路儲存系統上，以因應各種層面的錯誤。容錯技術可分為兩大類，一類是使用儲存多份複本資料來提供資料容錯能力，另一類則是使用了容錯編碼技術。由於使用容錯編碼技術的儲存本益比較好，許多先進的儲存系統都採取此技術。在使用容錯編碼技術的儲存系統中，資料先被容

錯編碼後分散儲存到系統中的各儲存伺服器中。然而容錯編碼技術只處理了維持整個儲存系統的不間斷服務，並沒有全然修復系統的功能。

(3) 資料完整性授權驗證機制

在使用者儲存在雲端的資料相當大量的情況下，便需要一個有效率的方法進行資料的完整性驗證來防止資料毀損等情況。最簡單的方法即是將資料從雲端下載回來和原始資料進行比對，但是這樣做不僅會耗費掉大量的頻寬，使用者也需耗費大量的空間來儲存原先的資料，如此一來，便失去了雲端儲存的優勢了，所以我們需要一個更有效率的方法來進行雲端資料的完整性驗證。另一方面，使用者有時候需要向他的親朋好友或是工作夥伴分享他的資料，被分享人也必須要能夠對資料進行完整性驗證的動作，這些資料只流通於他們之間，資料完整性驗證的能力也限定於他們之間擁有而已，這意味著使用者不僅僅是能夠授權驗證能力給他的親朋好友及工作夥伴，還必須能夠註銷授權出去的驗證能力，以及防止被授權人將驗證能力轉授權給他人，進而達到控制管理可授權的驗證能力。

- **基於機器碼之 Windows 惡意程式行為分析雲端平台**

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，對系統中的檔案或是網路上的封包進行掃瞄比對，對於已知的惡意程式與攻擊偵測十分準確且快速，但因為垃圾郵件相較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垃圾郵件成效不彰，所以需要重新設計一套分析架構，去收集大量的樣本並進行資料探勘以找出有用資訊，於是便想到利用雲端運算的架構來快速分析大量的資料，其目的主要在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。我們將整合修改先前開發的惡意檔案檢測模組來進行一系列分析。惡意檔案檢測系統要提供一個全面性的分析。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。由於動態分析能夠彌補靜態分析不具有執行狀況的資訊之缺點，所以本子系統將結合兩大分析技術，已達到較完整的檔案檢測分析。

本子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。第一點，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係的，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。第二點，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，公開雲端的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。

- **基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制**

雲端運算旨在透過網際網路來達到整合與遞送 IT 服務的功能。傳統上來說，各機關企業需要建立並維持自己的 IT 機房，或是使用所謂的主機代管服務。不管是哪一種作法，皆一定程度上會在硬體、網路、軟體及維護人力上重複投資。理想設計的情況下，IT 系統必須能夠承受尖峰時段的服務量，但這在離峰時段所閒置的硬體與人力卻會造成一定的浪費。對於中小企業而言，要營運一個 IT 中心事實上是非常沈重的負擔。而這也是為何往往很多中小企業的 IT 系統很難在功能性、穩定性與安全性上面面俱到的原因。

雲端運算希望將這些原本四散的 IT 資源整合在一起，成為一朵雲。當一個企業或機關需要使用 IT 的時候，可以透過網路來存取這朵雲所提供的 IT 服務。也因此，各個機關企業不再需要自己去建置機房，架設伺服器軟硬體、佈建昂貴的高速網路。透過雲端整合 IT 亦可以減少不必要的重複投資(比如說不同企業間的系統可以共享一個磁碟儲存陣列、共享 CPU 的運算時間)，達到更高的資源利用率。另一方面，假設這朵雲夠大，那麼提供雲服務的公司也就具有一定的經濟規模。他們也因此有足夠的資金可以投入系統性能優化、系統可靠性與安全性的提升，而不僅只是勉強符合基本的系統功能性需求而已。而這也是雲端運算相對於傳統 IT 架構的另一項優點。以目前而言，雲端運算服務可分為三個階層。最上層的是所謂的軟體服務雲 software as a service (SaaS)。第二種型態的雲是所謂的平台服務雲 platform as a service (PaaS)。第三種型態的雲是所謂的基礎建設服務雲 infrastructure as a service (IaaS)。

雲端毫無疑問是目前一個炙手可熱的話題。當然許多人心中一定會有的一個疑惑是如果台灣也要切入雲端，要如何著手？ 綜觀目前市場上的成功者，我們可以發覺他們一個共同的特點就是背後皆有強大的 infrastructure 在支撐。Google、Amazon、Salesforce 等在世界各地均佈有規模龐大的機房。也因此台灣若想在雲端上有所斬獲，或又純粹只是不想

受制於人（如直接使用 Amazon 或微軟所提供的雲），那我們勢必得發展出自己的 infrastructure。有鑑於台灣本身資源有限，顯然我們不可能指望有心從事雲端發展的企業皆各自去建設自己的 infrastructure(如 Google 有自己龐大的 infrastructure 在支撐自己其 search engine、Gmail、YouTube 等 SaaS)。也因此，發展 IaaS Cloud 以提供國內企業一個 infrastructure 應是我們的首要目標。

台灣的硬體能力舉世聞名。對於建置 IaaS 雲所需要的硬體我們有絕對的能力可以掌握，而這也可視為我們發展 IaaS 雲的優勢之一。再者以國家安全的角度而言，掌握自身 IT Infrastructure 亦是一個沒有討價還價空間餘地的議題。而這也是為何即便 Amazon EC2 已很成熟，微軟也在推他們的 Azure Cloud Platform，我們卻仍必需要投入力量在 IaaS Cloud 上面的研究與發展。

- **計與實作基於雲端技術之安全實驗觀測網路**

「雲端運算」又可細分為「雲端服務」與「雲端技術」。前者著重於透過網路連線取的遠端主機所提供的服務，如 Amazon EC2 是 SaaS 概念的實現。「雲端技術」則強調如何利用虛擬化及自動化技術來實現運算資源最佳化與資源共享等概念，例如利用虛擬機器 Xen 依使用者需求配置運算資源（運算核心與記憶體）。然而，目前的雲端技術卻尚不足以支援即時性（real-time）與仿真性（fediality），也很難滿足這些即時應用與仿真應用的需求。

以網路測試平台為例，研究人員或網路應用開發人員常需建置實驗網路，用此測試其新開發之產品或新提出技術的可行性與穩定度等。為了要提供高擴充性、可重置實驗、高仿真度的環境，在過去幾年，國內外知名大學紛紛投入研究分散式的網路架構與測試平台，包括 Emulab、ORBIT、APE（Ad Hoc Protocol Evaluation）、Roofnet、MiNT（Miniaturized Wireless Network Testbeds）、WHYNET（Wireless HYbrid NETwork）、Netbed、TESTBED@TWISC 等等。這些網路測試平台有些更延伸支援網路安全測試或無線網路測試，如 DETER、SWOON 等。

不論是哪一種測試平台，其設計宗旨與雲端運算不謀而合，都是期望將實驗所需的運算能力放置在遠端主機上，本機端僅透過瀏覽器或簡單的操控介面，即可配置、操控實驗，並讀取實驗結果。然而，以目前的雲端技術而言，雖然能夠隱藏平台硬體、作業系統的異質性，也能夠隔絕各運算叢集之間的干擾，但是，運算叢集的安全性、仿真性等卻不在考慮之內。

參考國內外目前雲端技術與網路測試平台之設計，本計劃設計了一套應用雲端技術之安全實驗觀測網路，使用者能在此雲端安全實驗觀測網路中，配置其所需之測試拓樸，並進行其實驗。以下稱所有配置之節點為「運算叢集」。除了應用現有雲端運算之虛擬化與自動化技術外，同時兼顧網路實驗與觀測時的必備條件，包括隔離性、仿真度、重複性、擴增性、資源共享、擴充性等。

- **隔離性（Isolation）：**

  預防運算叢集之外的封包有意或無意地對其內之測試節點造成影響；同時，也必須避免運算叢集中實驗用之惡意程式碼或病毒流出至雲端其他主機或外部公用網路。此外，亦必須考慮不同實驗之間的干擾問題，包括動態執行期間的記憶體資料干擾或執行結束後殘留記憶體之資料遭到側錄等問題。

- **仿真度（fiedality）：**

  與一般模擬測試不同，除了網路狀況之外，網路測試平台應能仿真（emulate）真實封包傳送接收情況，包括硬體瓶頸（CPU 使用率、緩衝區大小、網路卡速度等）。

- **重複性（Repeatability）：**

  建置網路測試平台的其中一個重要挑戰即為重現之前的實驗，包括當時的硬體配置、網路規模、拓樸、頻寬、狀況等等。

- **擴增性（Scalability）：**

  在真實網路系統中，網路規模將隨著使用量與使用者需求逐漸增大。為了能有效地仿真實際的網路系統，好的測試平台也應該支援規模的縮放，以期正確地取得測試效能或正確性。

- **擴充性（Extensibility）：**

  除了傳統的乙太網路之外，各種無線網路技術也逐漸興起，並普及於大眾。為了能提供使用者全方位的測試環境，測試平台應能支援各種不同網路技術的擴增，如 WiFi、WiMAX、無線感測網路等等。

參考上述各設計需求，本計劃所設計之雲端安全實驗觀測網路應具備下列特性：

- 解決實驗互相干擾問題。
- 解決系統安全問題，包括惡意程式碼外流或被駭客攻擊。
- 能仿真真實網路，包括其硬體、作業系統與網路配置。
- 能重複與重現各測試實驗。
- 雲端安全實驗觀測網路能支援未來的規模擴充，如增加實驗節點等。
- 支援更多新興網路技術（WiMAX、3.5G）之擴充。

二、 文獻探討與相關研究

本計畫包含四大研究主題，茲將各主題之相關文獻探討與目前研究狀況說明如下：

- **支援多樣功能之雲端資料安全儲存**

一般入侵偵測系統通常會使用「關鍵字比對」的技巧來分析和檢查系統資訊是否遭受到惡意的攻擊，然而這些方法都是直接使用原始資料(明文)進行比對。所以，當我們將這些入侵偵測系統移到雲端上來執行運作時，便有可能會造成使用者的隱私資料洩漏的問題。Song, Wanger, Perrig [14] 提出了一個用來解決關鍵字搜尋加密資料的方法。使用者將資料加密儲存在不可完全信任的遠端伺服器上，接者使用者可以透過關鍵字找回其對應的資料。

在針對容錯編碼分散式儲存系統設計的修復機制中，Dimakis 的研究是以修復單一錯誤為目標[17]。當系統中有一個儲存伺服器發生錯誤的時候，系統會新增一個儲存伺服器，新的伺服器會向其他伺服器索取資料，當正常運作的伺服器收到要求後，會針對自己儲存的資料進行處理之後再回傳給新伺服器，在新伺服器收到足夠多的資料時，便進行運算，最後將運算結果儲存起來。一個修復機制必須使得新的伺服器在功能上取代發生錯誤的伺服器。當時學者們提出在衡量修復機制的效率時，以每個儲存伺服器的儲存量與修復一個錯誤時所耗費的網路傳輸量來評估，爾後，有學者研究定義出兩個評量值之間的權衡關係(tradeoff)，並將其定義為一個曲線。一個修復機制的兩個衡量值若落在該曲線上，則被稱為 Regenerating code[18][19]。在修復機制中，若新增的儲存伺服器完全還原了發生錯誤的伺服器中儲存的資料，此類的修復方法稱為 exact repair，與之相對的方法則被稱為 functional repair。Rashmi 等學者提出 exact repair 的修復機制並且證明其效率衡量值落在 regenerating code 的曲線上[20]。Shah 等人則考慮網路狀況為不對稱的系統模型進行研究[21]。其他學者紛紛考慮不同的系統模型演化出不同修復機制[22][23][24]。若系統只能修復單一錯誤，那麼只要有錯誤發生，系統就要馬上進行修復，這對系統造成了過大的負擔。Hu 等人提出針對兩個以上錯誤進行的修復機制[25]，讓系統可以一次修復多個錯誤。當系統中出現 a 個錯誤時，就新增 a 個儲存伺服器，並讓 a 個伺服器詢問所有的伺服器取得既有的資料，再對得到的資料進行運算後儲存下來。Oggier 與 Datta 則提出了一個高效率可以修復多個錯誤的機制[26]，但是能夠應用的儲存系統必須是集中式的結構，也就是要有一個中央控制單位。在儲存系統發生錯誤的相關議題中，Dikaliotis 等學者考慮偵測資料錯誤(faulty error)的存在[27]。Rashmi 等人則提出可以結合兩個容錯編碼來提供更好的修復功能[28]。Pawar 等人討論當修復機制在執行的時候，網路上的竊聽者是否會破壞資料隱私性的問題[29]。Papailiopoulos 與 Dimakis 則證明了在最大化資料隱私性與最小化修復網路頻寬問題中間的對等關係[30]。無論是修復單一錯誤或者多個錯誤，我們發現到的是在效率評估的既有方法上，沒有考慮到兩個伺服器之間建立連線的成本，在網路傳輸資料總量相同時，若需要的連線數量較多，則建立連線的成本就會對傳輸成本造成影響。令系統中伺服器的總量為 n，既有的方法中，每修復一個錯誤都需要 n-1 個連線，我們認為這個成本有很大的改進空間。另一方面，我們特別針對非集中式容錯編碼儲存系統進行研究。此種儲存系統首先由 Dimakis 等學者提出，我們將資料隱私安全性加入儲存系統中，並在早期的研究中提出了具有安全性的非集中式容錯編碼儲存系統[32][33]。有別於一般的容錯編碼

儲存系統，非集中式的容錯編碼使得儲存系統不需要常設一個中央管理單位來維護系統全域參數，個別的儲存伺服器可以獨立進行編碼運算，無須彼此溝通，此結構很適合非集中式的環境，譬如點對點(P2P)環境或者是無線感測網路。

在雲端資料的完整性驗證的研究中，Ateniese 等人定義了 provable data possession（PDP）模型。他們提出了一套非對稱性密鑰的 PDP 方法，使用 homomorphic verifiable tag 來驗證使用者資料。根據 homomorphic verifiable 的特性，雲端儲存伺服器可以將使用者資料的 tag 線性組合起來，組合成單一個 tag 並回傳給使用者，以達到有效率的資料完整性驗證。後來，Ateniese 等人提出了一個對稱性密鑰的 PDP 方法，支持儲存資料的動態操作，像是插入刪除替換等。他們的方法具有可擴展性和高效率，然而，資料完整性驗證的次數卻是有限制的,取決於資料前處理時所插入的 token 數。Erway 等人提出的 dynamic provable data possession（DPDP）使用 rank-based skip list 來驗證資料完整性，skip list 是一種類似於 Merkle hash tree 的驗證用資料結構，DPDP 支援使用者資料上的動態操作。Juels 和 Kaliski 提出了 proof of retrievability（POR）模型。POR 確保儲存的資料可以被使用者正確地存取到，而 PDP 確保資料完好地儲存在雲端儲存伺服器中。Juels 和 Kaliski 的方法同樣需受限於資料前處理時插入的 token 數。後來，Shacham 和 Waters 提出了一個支援無限制檢驗次數的 POR 方法。Bowers 等人提出了一個理論框架去設計 POR 方法，該框架採用兩層的錯誤糾正碼（error correcting code）從雲端伺服器的回應中回復使用者的資料，他們的框架改善以往 POR 的研究成果。Wang 等人提出了支援資料動態操作和可公開驗證的 POR 方法，他們的方法同樣使用了 Merkle hash tree 資料結構以支援使用者資料的動態操作。為了同時實現高可用性和資料完整性檢查，使用者資料通常會有多個副本或是套用一些錯誤糾正的編碼方法。Curtmola 等人提出了 MR-PDP，確保每個獨特的副本都存在於儲存伺服器中。Curtmola 等人又提出了一個強大的資料完整性檢查方法，使用前向錯誤糾正碼（forward error correcting code）。後來，Bowers 等人提出了 HAIL，HAIL 提供了高可用性和資料完整性，HAIL 使用了多層的錯誤更正碼（error correcting code），可確保資料在分散式的儲存環境下，仍有一定的可用性。

- **基於機器碼之 Windows 惡意程式行為分析雲端平台**

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[52][53][55][56]。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[54]。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[61][62][63][65]，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 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[57][58][59][60]。

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[64][66][69]。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。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，預計能在計畫期間內前完成。

- **基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制**

IaaS 雲目前市場的領導者是 Amazon。Amazon 的 EC2 (Elastic Compute Cloud) 提供客戶透過網際網路配置虛擬機器[70]。

例如三家公司 A、B、C 分別在 EC2 上配置虛擬機器，底層是由四台實體機器所構成。每台實體機器上有安裝 Xen Hypervisor[71] 來提供虛擬化的服務。用戶可以在所配置到的虛擬機器上安裝各自所需的作業系統 (如 Linux、 OpenSolaris、Windows Server 等) 以及所需的應用服務程式（比如說資料庫系統、網站伺服器等）。用戶更可以自己去設定這些虛擬機器間的網路，以及配置各台機器所對應的實體 IP 位置等網路設定[72]。對於用戶來說，透過 Amazon EC2 所建立的虛擬機房所需的成本一般來說會比搭建實體機房來得更為便宜。另一方面，底層系統的可靠度以及安全性的部分亦可交給 Amazon 來負責處理。

IaaS 雲最關鍵的技術之一是機器以及網路的虛擬化。透過虛擬化，我們可以在一台實體機器上執行多個作業系統，也因此可以達到更高的底層硬體資源的利用率。另一方面虛擬化也讓系統佈建變得更為快速。虛擬化一般分為兩種類行。其一是直接在實體的機器上面裝一個 Hypervisor，然後可於上面配置多個虛擬機器，每個虛擬機器上面安裝各自的作業系統（Guest Operating System）。這稱為 Type-1 的虛擬化。如 VMWare ESX、Citrix Xen [74]、Microsoft Hyper-V[73]均屬於此類。另一種型態的虛擬化是在一台已經裝好作業系統 (Host OS)的機器上裝上一個 Hypervisor，然後再於 Hypervisor 內運行虛擬機器。Hypervisor 必須要透過 Host OS 的系統 API 才能跟底層硬體溝通。Type-2 Hypervisor 本身受到 Host OS 所加諸的一些限制，也因此在執行需要特殊權限的程式碼的時候，需要輔以 emulation 或 binary rewriting 等技巧來處理。Type-2 Hypervisor 包括了如 VMWare Fusion、VMWare Workstation、Microsoft Virtual PC、Oracle VirtualBox 、HXEN (Hosted Xen)等。對於 IaaS Cloud 來說，使用 Type-1 Virtualization 是一個比較常見的作法。主要原因不外乎因為系統的主要目的就是提供虛擬化環境，也因此底層先放個 host OS 也就顯得多此一舉，更遑論這對虛擬機器的執行性能會有所減損。

- **基於雲端技術之網路安全實驗觀測平台**

本系統將結合網路測試平台與雲端運算的虛擬化技術，意圖在單一台實驗主機上建立多台虛擬機器，並利用這些虛擬機器作為網路實驗中的實驗節點。因此，在本節中，我們將具體呈現現有雲端平台與網路測試平台的研究狀況。

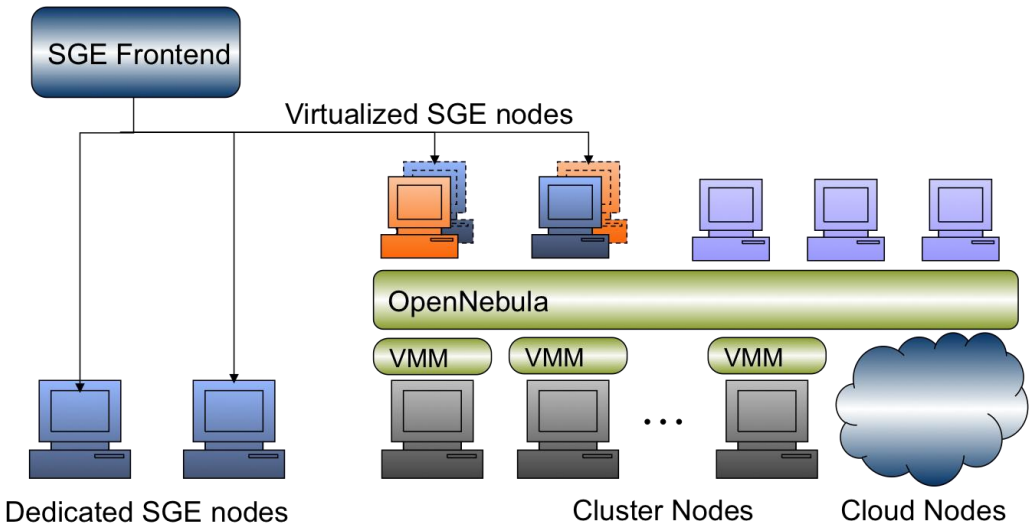
□ 雲端運算

本節將介紹目前較為知名的雲端服務與平台，包括 Amazon EC2 與微軟的 Windows Azure 平台。

■ Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2)是一種網頁服務，可以提供可變大小的計算能力。專門針對開發人員而設計，使其能輕易地調整網站計算規模。為了達到公平計費的目的，EC2 使用 Xen 虛擬化技術，依據所配置的處理器運算能力與記憶體大小，將其運算節點分為八大類，包括 m1.small、m1.large、m1.xlarge、m2.xlarge、m2.2xlarge、m2.4xlarge、c1.medium 以及 c2.xlarge。m1 類均配置一般記憶體，依照所配置的記憶體大小分為 small、large 與 xlarge。m2 類為高速記憶體，c2 類節點則配有高速運算能力之處理器。每種節點上均可選用不同的作業系統（Linux/UNIX、Windows），並可依據資料庫需求，增配 SQL 伺服器。此外，EC2 服務也能依據使用者對 IP 位址、負載平衡以及資料傳輸量的需求，適度地提供不同的配置，並依使用情況收取不同的費用。當使用或要求的資源少時，收費也會跟著減少。

簡單來說，Amazon EC2 簡單的網頁服務介面可以讓使用者在最少衝突的情況下，取得與配置所需的計算能力。同時，EC2 也允許其使用者可以全面控制這些計算資源，並於 Amazon 所認可的計算環境中執行應用程式。EC2 可以將取得新伺服器與開機的時間縮減至數分鐘，讓使用者可以隨著計算需求改變時，快速隨狀況調整運算能力。此外，EC2 也提供開發人員一些工具，開發各種可以由錯誤中回復的應用程式，並使這些應用程式可以免於遇到常見的錯誤情況。

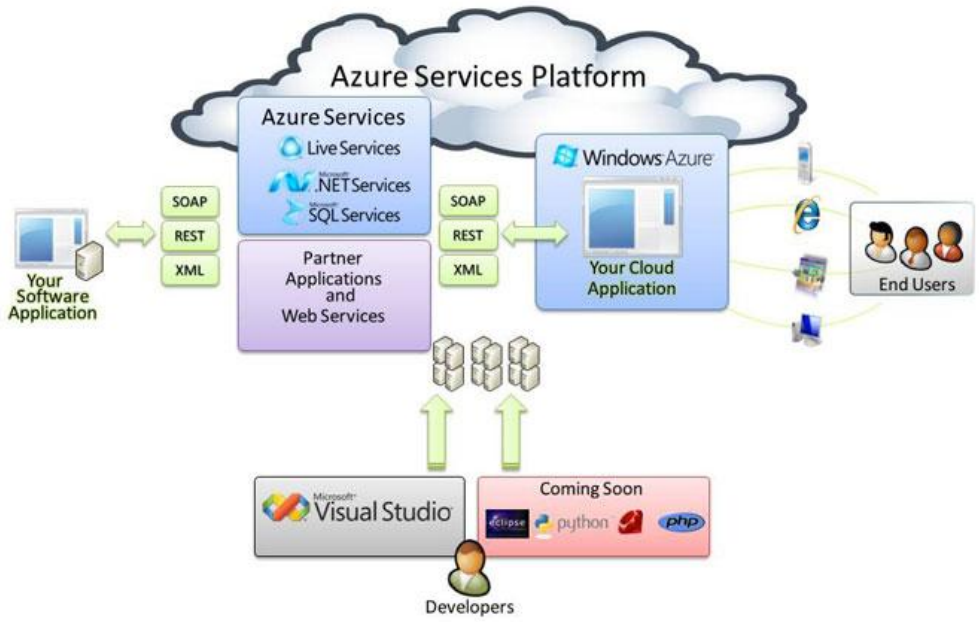


圖表一：Amazon EC2 之架構

■ Microsoft Azure Platform

一直以來，微軟都是以本機端的視窗軟體為其根基。然而，這卻與雲端運算的宗旨（不需安裝額外的本機端軟體，所有資源都在雲端，本機端只需能連上雲端的設備與瀏覽程式即可）相違背。為了因應雲端的流行趨勢，不落人後的微軟也開始推出免費的雲端服務，搭配其本機端軟體，可以讓使用者在熟悉的環境中，享受這些免費的雲端服務。

Windows Azure Platform 是微軟新推出的雲端服務平台(含有 Windows Azure 作業系統、.NET 服務與 SQL 資料庫)，可提供開發人員一套更有彈性的視窗環境，使其能快速地開發雲端應用與服務。Windows Azure 是一套雲端服務的作業系統，除了開發之外，也能提供管理服務的機制與環境，開發人員可以透過 Azure 平台提供使用者所需要的計算與儲存能力，甚至能透過網路管理其網站應用。由於 Windows Azure 整合了 Visual Studio，也支援多種標準與協定，如 SOAP、XML、PHP 等，因此，能使開發人員過去的開發經驗繼續傳承與延伸。



圖表二：Windows Azure Platform 之架構

□ 網路測試平台

在網路測試平台方面，本節主要介紹目前較為知名的幾個國內外測試平台，包括 Emulab、DETER、TESTBED@TWISC 等。

■ Emulab

Emulab 是猶他大學所研發設計之網路測試平台，利用分散式系統與網路建置出一套研究用的仿真平台。在此平台中，使用者可以利用 NS 語法，將平台內的各實驗節點任意連接成一個網路拓樸。平台可以支援多組實驗同時進行，每個實驗網路均以 VLAN 隔開。 不管實際接線情況為何，配置在同一個 VLAN 下的網路

拓樸彼此之間可以互相通訊，就好似連接在同一個區網之內。此隔絕性可以保證各實驗之間彼此不會互相干擾。在 Emulab 中，使用者建置新實驗的步驟包括： 1) 配置實驗節點；2) 配置 VLAN，以建構所需的拓樸；3) 將映像檔載入所指定的實驗節點中。然後，使用者就可以開始進行其測試實驗。

圖表三：猶他大學的 EmuLab 平台

- DETER

以 Emulab 為基礎，DETER 測試平台更進一步地提供安全保護機制，避免測試平台本身遭到外部攻擊，同時也保證測試平台內部的實驗資料（尤其是含有惡意程式碼的實驗）不會流入外部公用網路。此平台的特色在於提供「安全」的測試環境。其設計已經整合於目前的 Emulab 中，並有眾多使用者以此平台作為測試其新方法的依據。

圖表四：柏克萊大學的 DETER 平台

- TESTBED@TWISC

Testbed@TWISC 由國內成功大學所建置之網路測試平台。此平台以猶他大學所授權的 Emulab 為基礎，具備網路測試所需的隔離性、仿真度等特性。由於國內外硬體規格差異，成功大學研發團隊在取得 Emulab 授權之後，便積極尋找可替代之國產硬體（包括實體實驗節點、電源控制器、交換器等設備），並自行研發相關軟、韌體，利用簡單的網頁操作介面，以因應國內資訊安全研究與實驗之測試所

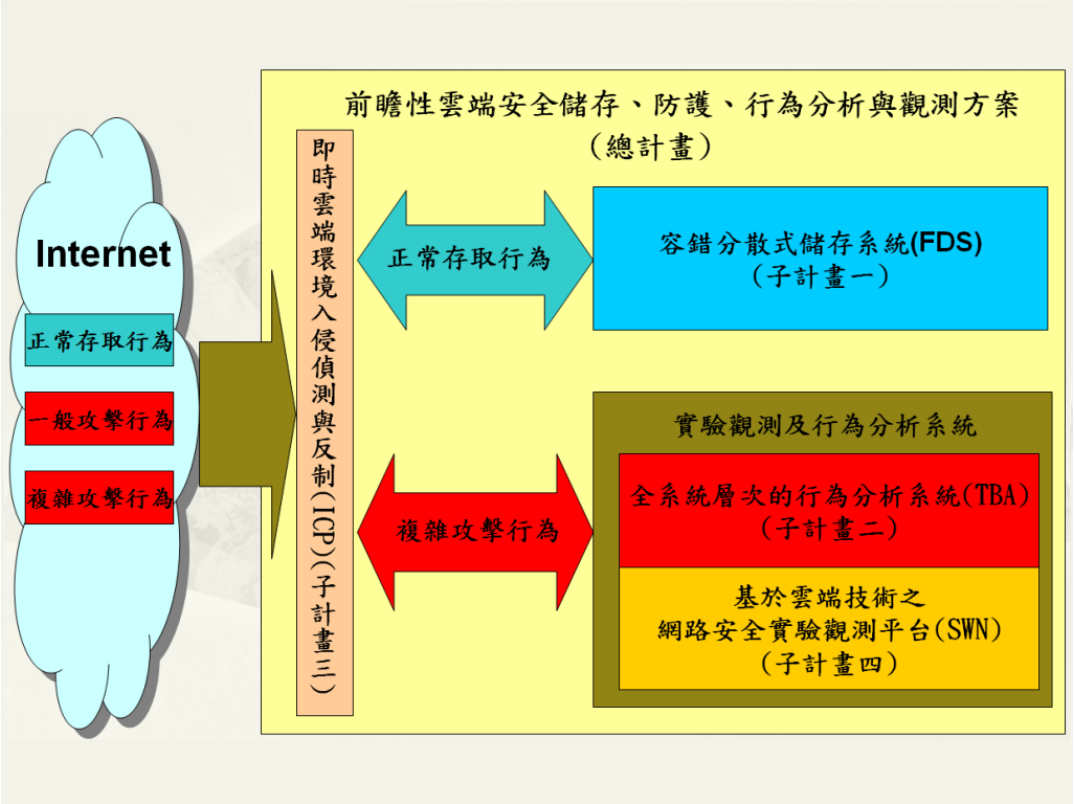
需。此測試平台中的實驗節點皆為實體機器，如前所述，使用者進行實驗之前，必須先配置得實驗節點。爾後，使用者便可以任意控制整個實驗節點（實體機器），包括 root 權限、作業系統與軟體安裝等。

三、 計畫目的

本技術為一安全雲端平台建構方案，涵蓋系統安全、儲存安全、網路安全、與安全測試等面向，具體可分為以下四個子系統：

- 多功能容錯分散式儲存系統
- 全系統層次的惡意程式行為分析
- 即時雲端環境入侵偵測與反制
- 基於雲端技術之網路安全實驗觀測平台

所有來自於 Internet 的行為大致上可分為正常存取行為、一般攻擊行為與複雜攻擊行為三種，這三類行為都會先經過即時雲端環境入侵偵測與反制系統的檢查。其中，屬於一般攻擊的類型都會在這一個階段被擋下，而正常的存取行為會被導向到容錯分散式儲存系統。疑似為複雜攻擊的程式，可輸入至全系統層次行為分析系統，進行惡意程式行為分析。此外，可疑的惡意網路封包亦可利用安全實驗觀測平台進行隔離實驗與觀測。本系統之架構如下圖：



圖表五：系統架構圖

茲將各子計畫之具體研究目的分述如下：

- **支援多樣功能之雲端資料安全儲存**

本計畫的目的為在考量使用者資料隱私性以及雲端儲存資料安全性下，發展支援多樣功能之雲端資料安全儲存機制。此部份的研究目的可分為以下三項：

- **保護隱私的雲端入侵偵測**

此項目的目的是希望提供一個模式來保護使用者上傳資訊的隱私且依然能夠將這些保護過的資訊交給雲端來處理。我們將這個方法應用在入侵偵測系統中。藉此方法達到保護上傳的資訊的隱私且又能夠提供入侵偵測的能力。

☐ **非集中式雲端儲存系統中系統修復機制**

此項目的目的是為 (n,k) 分散式容錯編碼分散式儲存系統提供具有更好效率的修復機制，且該機制能一次修復多個錯誤。

☐ **資料完整性授權驗證機制**

此項目的目的是希望能提出一個可授權驗證的資料完整性檢查方法，進行驗證時不需下載完整的使用者資料，使用者也可以將驗證能力授權給信賴的代理人，進行授權的資料完整性驗證。另一方面，使用者還可以防止被授權人轉授權驗證能力給他人，或是註銷已經授權的驗證能力，達到控制管理可授權的驗證能力。

● **基於機器碼之 Windows 惡意程式行為分析雲端平台**

本研究執行之目的有二：(一) 提供雲端服務平台檔案之安全分析 (二) 開發與設計雲端架構分析平台以提高安全分析產能。以下對此兩大目的加以說明：

☐ **提供雲端服務平台檔案之安全分析**

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動態汙染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。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**

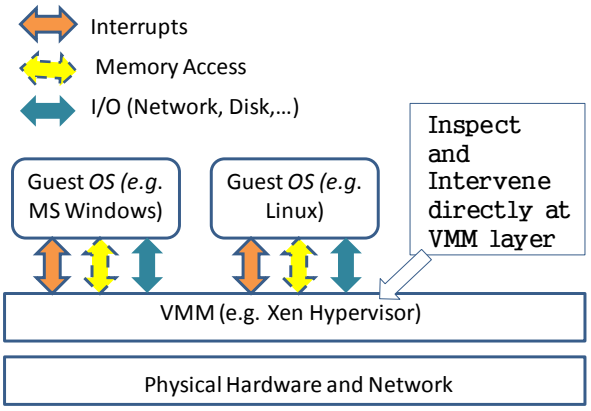
檔案分析的方式眾多，越來越多不同面向的分析工具大量的產出。但因為分析的方法不同，而有互補的作用。靜態分析與動態分析各有其優缺點，靜態分析雖然運算負擔較動態分析小，但卻無法得知執行時期的資訊，所以在分析應用上較適合特徵式偵測。另一方面，動態分析除了負擔較大的運算開銷以外，分析範圍需要大量的運算時間搭配，故並沒有一種方法是完美的。如果能夠簡易的整合市面上各種分析工具而達到互補作用，又可有效且平均的分配至多台電腦，則可以加強分析能力又可縮短分析時間。

為了增加分析的效率與產出，本研究計畫的第三項大目標，就是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：可擴充性、可自動化、與高容錯性。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。

- **基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制**

本研究的主要要目的旨在探討 IaaS Cloud 中的入侵偵測（IDS）[75]以及入侵防制/反制（IPS/IRS）[76]技術的研究與開發。對於保護 infrastructure，使用入侵偵測與反制系統是行之有年的作法。具體而言如一般電腦上所安裝的防毒軟體便是一個入侵偵測與反制系統的例子(其目的是偵測病毒並將其阻擋)。另外佈建所謂的網路安全閘道來即時監控含有攻擊性內容的封包與網路連線並予以攔截亦是入侵偵測與反制的另一個例子。然而當我們面對 IaaS Cloud，所映入眼前的景象已不再是我們所熟悉的實體主機、網路線、路由器、網路交換機等設備。在眼前的是透過 Internet 存取的一朵雲。這朵雲背後的機房可能遠在美國、遠在歐洲。以傳統系統管理者角度而言，一個顯然的疑問是我要怎麼去一朵虛擬的雲上面安裝網路安全匝道器？另一個顯然的問題是一個公司會考慮選擇使用 IaaS Cloud，而非自建資料中心機房莫過於是為了降低成本。倘若在資安支出上，採用雲計算所需付出的資安成本是一樣或甚至更高的話，那勢必就會降低 IaaS Cloud 的吸引力。



圖表六：透過 VMM 來達到偵測跟反制

本研究的主要構想如上圖所示。傳統而言，IDS/IPS 有兩種配置，選擇之一是裝在個別機器上面（上圖 Guest OS 上），另一個選擇是裝在網路上（上圖中的 Physical Hardware and Network，比如說所謂的網路安全匝道器即是採用後者這種配置方法）。然而由上圖可看出在 Xen 虛擬化後的環境中，Guest OS 的所有 I/O、中斷、記憶體存取均會透過 Virtual Machine Monitor (VMM)，亦即會透過 Xen Hypervisor 這一層。也因此對於 IDS 所需要的偵測功能以及 IPS 所需要的反制動作功能，均可以移至 VMM 層中。這樣子顯然的好處是我們不再需要於 Guest OS 中安裝如防毒軟體之類的工具，亦

不需要在實體網路上佈建網路安全匝道器。此構想最早是由 Stanford 大學的 Mendel Rosenblum 教授提出[77]。透過 VMM 監控上層 Guest OS 是一個非常重要且具實際應用價值的技術。目前仍存在的障礙是只能對底層(實體層)的記憶體、I/O 資訊的存取。也就是說 VMM 監控所看到的是底層的低階、缺乏結構性的狀態或事件，這與上層應用中的狀態或事件間存一個 semantic gap，直觀地來說，由於 Hypervisor 掌控了虛擬機器對底層硬體包括 CPU、記憶體、磁碟機、網路、周邊設備等的存取，所以我們可以透過 Hypervisor 來達到傳統 IDS/IPS 所進行的偵測、防護以及制動。

- **基於雲端技術之網路安全實驗觀測平台**

本研究著重於將虛擬機技術導入網路安全實驗觀測平台之建構，其具體研究方向如下：

- **虛擬機資源管理分析：**

  此技術將虛擬機技術導入網路測試平台，可依照各別的測試需求自動建立虛擬節點與網路連結。

- **虛擬機安全性分析技術：**

  此技術可確保虛擬節點之間的隔離性，防止運行中的惡意程式污染其他虛擬節點或是測試平台。

- **虛擬機安全資源管理機制：**

  此技術可建立自動化之虛擬機管理機制，測試平台可依照各別測試的需求管理各虛擬機之資源。

四、 研究方法

本計畫的組成結構如下:

<table>
<tr><td colspan="4">總計畫：前瞻性雲端安全儲存、防護、行為分析與觀測平台
主持人:曾文貴
共同主持人：謝續平、黃育綸、吳育松</td></tr>
<tr><td colspan="2">安全儲存與即時保護系統</td><td colspan="2">實驗觀測及行為分析系統</td></tr>
<tr><td>子系統

「子計畫一：支援多樣功能之雲端資料安全儲存」</td><td>子系統

「子計畫三:基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制」</td><td>子系統

「子計畫二：基於機器碼之 Windows 惡意程式行為分析雲端平台」</td><td>子系統

「子計畫四：設計與實作基於雲端技術之安全實驗觀測網路」</td></tr>
<tr><td>主持人:曾文貴教授</td><td>主持人:吳育松教授</td><td>主持人:謝續平教授</td><td>主持人:黃育綸教授</td></tr>
</table>

圖表七：子系統結構表

總計畫已整合上述四個子計畫的研究成果，並已發展出一個安全雲端服務平台建構方法。在執行面上總計畫負責跨子計畫間的溝通機制，並透過定期會議以及成果查驗確定各子計畫之研究時程與成果皆符合原提案之規劃。

此外，總計畫亦透過建立人力資源表以確保各子計畫之人力分配符合需求。在設備資源方面，所有設備資源依照「專用資源」以及「共用資源」的差異先行分類。將具有特殊功能以及高度專用性的設備或資源編入「專用資源」類，其他不具有特殊功能以及高可共用性的設備或資源編入「共用資源」類。在計畫執行中致力提高屬於「共用資源」類的比例以及分享程度，這個做法確實能提高資源整合的程度，並同時提高整體資源運作效率。

針對各子計畫陸續產出之具體成果，總計畫按照整體規劃建構適合的測試計畫或成果驗證計畫，以及準備所需的測試器材。同時亦將針對各子計畫之研究成果進行統合測試及成果驗證，以確定所有實體系統能夠正確運行、且所有演算法及協定能達到預期的功能、效率、以及安全強度。

綜合以上所述，本整合計畫之研究方法規畫如下:

- **計畫主題整合與計畫資源管理**

本計畫包含四個研究子計畫，各子計畫由總計畫負責協調各項研究進度與人力物力資源之調配，其具體內容說明如下：

- **規劃整體計畫研究架構：**

  總計畫已在計畫初期設計完成整合系統的完整架構，特別是與子系統間之功能性的畫分與資料流的傳遞方向。

- **建立各子計畫經驗交流機制：**

各子計畫人員各有不同的專長背景，總計畫執行過程中常態性的舉辦經驗交流與進度會議，可縮短整體計畫團隊學習時間的效果。

- **建立各子計畫設備資源共享機制：**

  計畫設備資源有限，總計畫已將各子計畫設備資源分類，並增加資源的可共用性，可最大化所有設備的使用效率。

- **協調整合系統之基礎軟硬體架構：**

  總計畫向各子計畫調查軟硬體需求，並協調所需要的整合式平台，避免可能的系統衝突，以確保研究成果之整體性。

- **撰寫總成果報告：**

  總成果報告除了實際系統的各種說明文件以外，也包含所有相關的發表論文，最後並把整體系統設計理念文件化，以便相關研究者可以在這個基礎上繼續更深入的研究。

各子計畫之個別研究方法規畫如下:

- **支援多樣功能之雲端資料安全儲存**

  本計畫的研究方法可依研究成果分成以下三項分別說明：

  - **保護隱私的雲端入侵偵測**

    我們採用前面所提到的 Song, Wanger, Perrig 之關鍵字搜尋加密資料的方法來保護使用者上傳資訊的隱私。我們利用候選者名單來讓雲端提供者無法確定使用者上傳的資訊確切為何，其中加密金鑰雜湊部分的長度會影響到候選者名單的大小，藉由調整加密金鑰雜奏部分的長度來改變候選者名單大小來達到保護隱私能力的強度。除此之外，我們會先將事先定義好的關鍵字放在雲端上，使用者只要上傳可疑的資訊到雲上，則我們會比對這資訊和事先定義好的攻擊來判斷使用者是否有遭受到惡意攻擊。在系統實作上我們以 Linux 來當作我們系統的平台，使用 Hadoop 來建立雲端環境，並利用 SNORT 來建立出我們要比對的關鍵字資料庫。藉由上述的技巧及環境來開發我們的雲端安全服務：Privacy-Preserving Cloud IDS。總括來說，我們利用下表中所列出的工具來建立我們實驗的環境，最後根據這個環境做隱私、偵測率和效能的評估和實驗。

	使用工具清單
雲端環境	Ubuntu + Hadoop + Java + OpenSSH
關鍵字資料庫	SNORT rule
保護隱私方法	jpcap library + client-server program

圖表八：使用工具清單

本系統分為兩大部分：使用者端和服務者端。使用者端是一個 java 程式，用來收集使用者電腦的資訊，經過隱私處理後，傳到雲端服務上做分析處理，在做最後的判斷，來判斷使用者的電腦是否安全無疑。服務者端採用 MapReduce 的程式架構來完成 hidden keyword search 的功能，用來比對使用者的上傳。除此之外，針對比對的資料庫，我們事先做好隱私處理，將這些資料放在 Datanode 上(HDFS)，用來做比對處理

□ 非集中式雲端儲存系統中系統修復機制

在雲端儲存系統中我們使用了容錯編碼來建構儲存系統的容錯能力，使用容錯編碼的儲存系統可以在系統中部分儲存伺服器無法提供服務時，維持整體儲存系統的對外資料存取服務。

經過我們對相關研究的資料收集與彙整後，我們發現現有的方法中，大多僅考慮一個儲存伺服器錯誤的狀況，並在錯誤發生後立即進行修復。有別於現有的方法，我們希望系統能夠先容許一小部份的伺服器錯誤，預測一個周期時間長度，進行周期性修復，以期避免經常執行大規模的修復動作。我們考慮當系統中有 x 個伺服器錯誤時，使用 x 個新的伺服器加入系統並進行修復動作。這些新的伺服器可以索取舊伺服器中儲存的資料並進行相關計算，最後儲存一個計算後的資料，一個修復後的儲存系統必須維持夠高的資料存取成功機率。

我們探討了系統中可以承受的錯誤比率，一個新伺服器需要詢問的舊伺服器數量，修復一個錯誤需要耗費的網路傳輸資料量，以及每個伺服器的儲存量。目前主要的發現是，考慮一個有 n 個伺服器的儲存系統，其中任意 k 個儲存伺服器可以協助使用者將資料取回。隨著系統中的伺服器發生錯誤，在系統中仍正常運作的伺服器數量若能維持在足夠的數量下，一個新的伺服器可以向少於 k 個舊伺服器索取資料，並修復一個伺服器錯誤造成的損失。

為了能針對 (n,k) 分散式容錯編碼分散式儲存系統提供具有更好效率的修復機制(且能一次修復多個錯誤)。在一個具有 n 個伺服器的系統中，我們透過隨機程序的方法，使新增(1-α)n 伺服器能夠詢問隨機 u 個運作中的伺服器來進行修復，其中 $\alpha<1$。我們考慮了一個非集中式修復機制，讓每個新的儲存伺服器可以獨立的進行修復作業，無須彼此溝通。在我們的機制中 u 值可以遠小於(n-1)，同時我們的方法維持了非集中式系統結構的優點。為了更凸顯提出方法的效率，我們進行了數值與參數的分析與比較。就比較結果上來看，非集中式修復機制能夠具有更好的修復效率。

□ 資料完整性授權驗證機制

在本研究中，我們的資料完整性授權驗證，包含以下三種角色：使用者（Data owner）、代理人（Delegated verifier）、以及儲存伺服器（Storage server）。我們的設計以私密驗證為基礎，利用額外的代理金鑰將使用者驗證標記，轉換為代理人所能驗證的標記，以達到授權驗證之目的。使用者對資料產生驗證使用的標記，並將資料及標記一併上傳至儲存伺服器，之後，也將代理人的代理金鑰送給儲存伺服器；代理人可對遠端伺服器進行資料完整性的驗證，藉由傳送挑戰值，儲存伺服器可利用代理金鑰產生對應的完整性證明，代理人將可利用自身的私密金鑰驗證回傳的完整性證明。在設計上，我們利用 bilinear map 進行 Tags 的轉換，儲存伺服器使用代理金鑰將使用者的 Tag 轉換為代理人的 Tag，代理人可對此 Tag 進行驗證。上述方法可支援同時授權多數的代理人，而不會造成使用者的計算負擔，以及儲存伺服器的儲存負擔，每增加一位代理人，系統上只會增加一把代理金鑰，使用者只需要利用自己的私密金鑰去產生代理金鑰，不需要去存取在儲存伺服器上的資料及代理金鑰，授權的動作將十分簡潔。

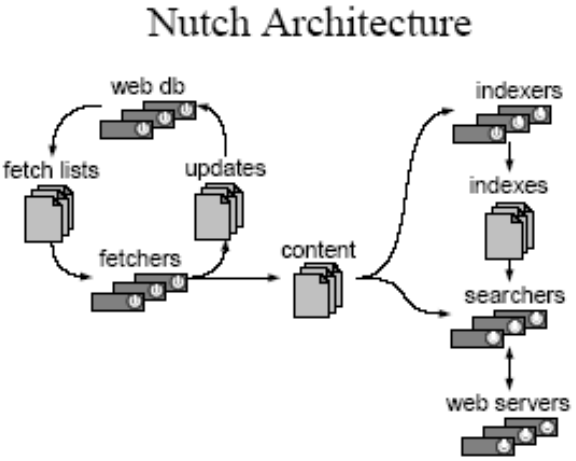
- **基於機器碼之 Windows 惡意程式行為分析雲端平台**

本分析平台主要由三個部分所組成：檔案來源子系統、儲存子系統、以及分析器管理子系統，以下將分部介紹之：

- **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。



圖表九：Nutch 系統架構

上圖為 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（上圖左半部）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。而 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：

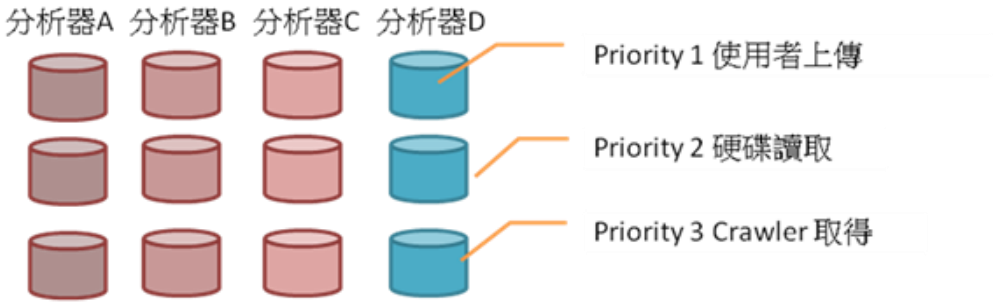
- **讀寫快速**

  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。

- **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。將在下面詳述。

本子系統在 Redis 部份實作了工作優先權佇列，架構如圖表十，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。



圖表十：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，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

圖表十一：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，如圖表十一。 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(TBA)**

本分析系統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被鉅細靡底的列出。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，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)：**

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，此實作系統針對兩個方向進行分析：(一)是否有新增的程序、(二)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。

- **偵測檔案修改行為(FLM)：**

運行具內核權限的惡意樣本(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。

- **偵測登錄機碼修改行為(REG)：**

  此子系統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，內核權限的惡意樣本還會竄改 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。

- **偵測網路通訊修改行為(NET)：**

  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通訊阜，這些新增的通訊阜即是該程序所開啟的。

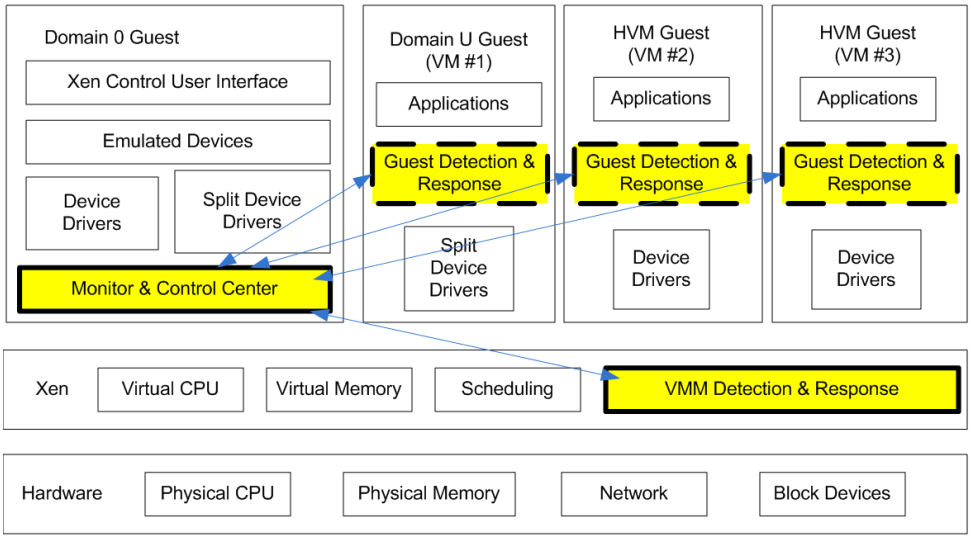
- **偵測驅動程式植入行為(DRV)：**

  為找出程式是否會動態載入可疑的驅動程式，本子系統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，可把可疑的惡意驅動程式找出。

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。

本系統採用的動態污染分析具有高度的研究價值與應用性，尤其針對現今多變的攻擊與惡意程式，必需依賴此技術以準確分析出它們的動態執行時期行為，以補足傳統分析方法的弱點。所謂的動態分析乃透過實際運行程式以獲得更多更深入執行時期的動態資訊，而動態汙染源分析乃眾多動態分析技術之一。在進行動態分析時，分析者必須提供一虛擬的運行環境，並在此虛擬出的環境中執行欲分析的目標。這樣的虛擬運行環境一來使分析者能完全掌握目標程式的執行過程，並使目標程式與真正的主機隔離。因此，為了大量地建置該分析系統，本計畫需要建立一虛擬環境以利進行分析。這也是本子計畫的規劃原因，在第一年度實作一個可擴充的惡意程式分析平台，而在第二年度加強其平台的分析能力。

- **基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制(ICP)**



圖表十二：Iaas Cloud 實驗平台(ICP)架構圖

本研究的目的著眼於將入侵偵測、入侵防護與反制的功能整合進 Xen Hypervisor 中。另外在 Domain 0 Guest 中我們會製作一個 Monitor & Control Center 來統籌該台實體機器上之虛擬環境整體的入侵偵測與反制工作。再者，如有須要，我們仍舊可透過於 Domain U Guest 中植入 Agent 的做法來達到使用方式上雷同於傳統 IDS/IPS 的入侵偵測與反制。

□ 系統呼叫之攔截

系統呼叫攔截是一個廣為被傳統 IPS 系統所使用來提供即時保護 (real-time protection)的一個技巧。比如防毒軟體會攔截 Windows 系統呼叫 *CreateProcess()*以達到在一個程式即將被執行前，先對其進行病毒碼掃描，並在當發覺該程式執行檔受到病毒感染時，能阻擋該程式的執行(*對 CreateProcess()回傳錯誤代碼 0*)，以即進一步對該檔案進行隔離(比如透過攔截 *CreateFile()*系統呼叫)。

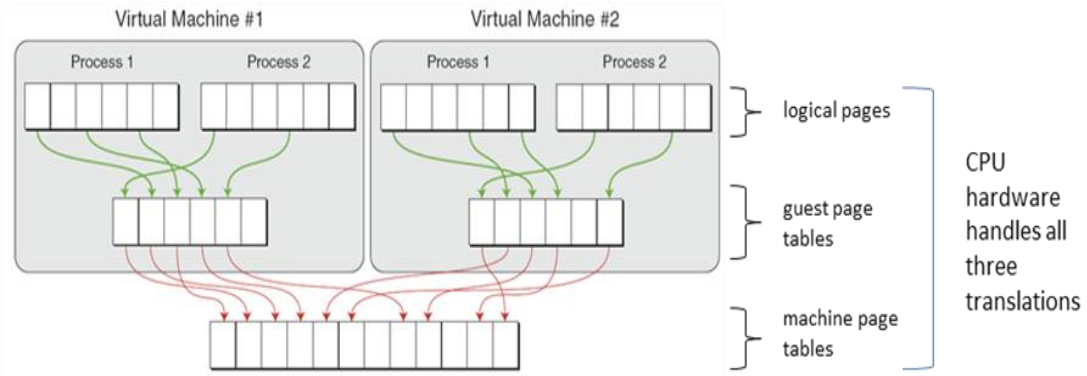
系統呼叫在傳統上是透過呼叫特定中斷(如在 x86 平台上 Windows 慣用 INT 0x2E，FreeBSD、Linux 慣用 INT 0x80)並透過堆疊或暫存器來傳遞參數。近期的作業系統已開始使用如 SYSCALL (AMD 架構) 或 SYSENTER(INTEL 架構) 等專門用來執行系統呼叫的指令集來取代較為耗時的中斷向量呼叫法。此一做法的好處是直接把系統呼叫處理函數的位置存在暫存器 MSR_LSTAR(AMD 架構)或 SYSENTER_EIP_MSR(INTEL 架構)中，然後透過 SYSCALL 或 SYSENTER 指令一步到位，免除需要存取 IDT，GDT 等相當耗時的記憶體存取。為了最大化虛擬平台上的執行效能，Xen Hypervisor 會儘可能地用硬體直接去執行上層的 Guest VM。而這也包括了對於絕大部分系統呼叫的函式碼亦是如此，也因此必須透過一些特殊的修改才能夠達到偵測與攔截 guest VM 內部系統呼叫的目的。

首先針對 INT 0x2E（Linux 或 FreeBSD 是使用 int 0x80）中斷叫用系統呼叫處理函數的此一情況，我們的作法是在 Xen 對 IDT 表格作初始化的過程的時候先將 0x2E 所對應的表格位置中的記憶體地址填入指向一個不存在的記憶體分頁中的地址。如此一

來，在系統呼叫發生的時候的前一剎那，會產生一個 page fault。這會導致 CPU 將執行權從 guest VM 轉回至 Hypervisor。如此便能攔截透過 INT 0x2E 進行的系統呼叫[80]。

另一方面，假若系統是使用 SYSENTER 或 SYSCALL 等指令來進行系統呼叫的叫用的情況，由於這兩個指令會直接跳至 Ring 0。如果是 PV Guest，這會導致一個 General Protection Fault[93]，執行權也就自動會轉進 Hypervisor（因為 guest kernel 不在 Ring 0)，也因此可以輕易攔截住系統呼叫。

如果是在 HVM(Hybrid VM with Hardware-assist)的 Guest 底下使用 SYSENTER 或 SYSCALL，由於 Guest Kernel 本身就是位於 Ring 0，也因此並不會造成 General Protection Fault。在 Ether[80]中，他們提出一個解決方法是將 SYSENTER_EIP_MSR 設向一個不存在的 page 而導致 page fault，然後執行權自動又會轉進 Hypervisor，如此便能攔截到系統呼叫。



圖表十三：Intel EPT

4.0 版的 Xen 開始支援所謂的 Intel Extended Page Table (EPT) [96] 或 AMD RVI/NPT [94]這些硬體對於虛擬 page tables 上的支援。因此，當 Hypervisor 配置好 machine page tables，將各 Guest VM 分配至不同區段的記憶體空間後，剩下的就完全交給 Guest VM 以及硬體去處理。

針對這種狀況，我們提出的方式是使用一般 Guest VM 無法執行特權指令的特性，將特定的特權指令塞到 Guest VM 處理系統呼叫函式的記憶體位置之前，因為 Guest VM 並不是在 root mode 執行，所以一旦嘗試要執行特權指令時，就會因為權限不足而產生 VMExit 進入 Hypervisor，而此時控制權就交到了 Hypervisor 的手裡，因此能夠達到攔截系統呼叫的功能。

但是在 Linux kernel 中，我們不能保證系統呼叫函式所在的記憶體位置前有足夠的空間來放置我們所要插入的特權指令，為了不影響 Guest VM 系統呼叫的正常執行並且成功塞入特權指令到系統呼叫處理函式之前，我們的做法是去修改 Guest VM 的 kernel，在處理系統呼叫函式之前插入兩個 nop 的指令，產生兩個 byte 的額外空間，以便我們之後插入特權指令(0x0fa2)。

□ 系統記憶體內容物之定位

傳統 IDS/IPS 是與 Host OS 共處的形式存在的，其在攔截系統呼叫後可以透過 OS 所提供的 API 如 GetProcessImageFileName、ReadProcessMemory 、OpenProcess 等方

式來讀取正在嘗試被攔截之系統呼叫背後的 Process 程式之詳細資訊以及其記憶體內容。比如說傳統防毒軟體在偵測到檔案即將被執行時，可對甫被載入記憶體的執行檔映像進行病毒碼的掃描。另一方面對於 Rootkit 的偵測，亦需要具備對於系統核心記憶體空間的掃描與解析能力[97-98]。

然而，對於在虛擬機器層要直接對上層的 Guest VM 之記憶體進行解析並不是一件容易的事情，因為我們不能透過 Guest VM 所提供的 API 如 ReadProcessMemory 來達到此一目的，因此我們利用在 Guest VM 內安裝 driver 的方式，透過 driver 列出各個 process 之詳細資訊，如執行檔名稱、位置以及 process 完整的記憶體區段，再加上 Xen 本身有提供讀取 Guest VM 記憶體空間的功能，我們只需要在控制中心內呼叫 Xen 所提供的記憶體存取 API(xc_map_foreign_range)，指定 Guest VM 的 domain id 以及想要存取的 Guest 實體記憶體頁面編號之後，Hypervisor 就會利用 EPT 將 Guest 實體記憶體頁面編號轉換成真正的實體記憶體頁面編號，接著就可以使用實體記憶體頁面編號來存取 Guest VM 記憶體內容。

取得 Guest VM 記憶體內容後，我們透過 disassembler 對記憶體進行進一步的解析，經由反組譯將記憶體內容翻譯成 high level information 之後，我們再檢查記憶體內容中是否有可疑的指令(一串 nop 指令附帶惡意程式碼)，或是不合理的程式流程轉移。另一方面，IDS/IPS 會需要對於記憶體空間有即時的監控能力。也就是說需要能即時掌握記憶體中的資料變化(比如說某個 process 突然動態連結一個新的 DLL 函式庫，IDS 得立即對他進行掃描)。

解析完記憶體區塊內容後，便可根據解析的結果來決定 process 記憶體區段中每個 memory page 在 EPT Entry 上的權限，透過權限管理的方式讓惡意軟體無法使用記憶體資源，藉此阻止惡意軟體的執行。為了達到這個目的，首先我們利用 Xen API(ept_walk_table)找到該 Guest VM 的 EPT 以及每一個 memory page 所對應到的 EPT Entry，再透過修改 EPT Entry 權限的 API(ept_set_entry)，便可限制 Guest VM 對 memory pages 的存取。

□ 檔案以及磁區之監控

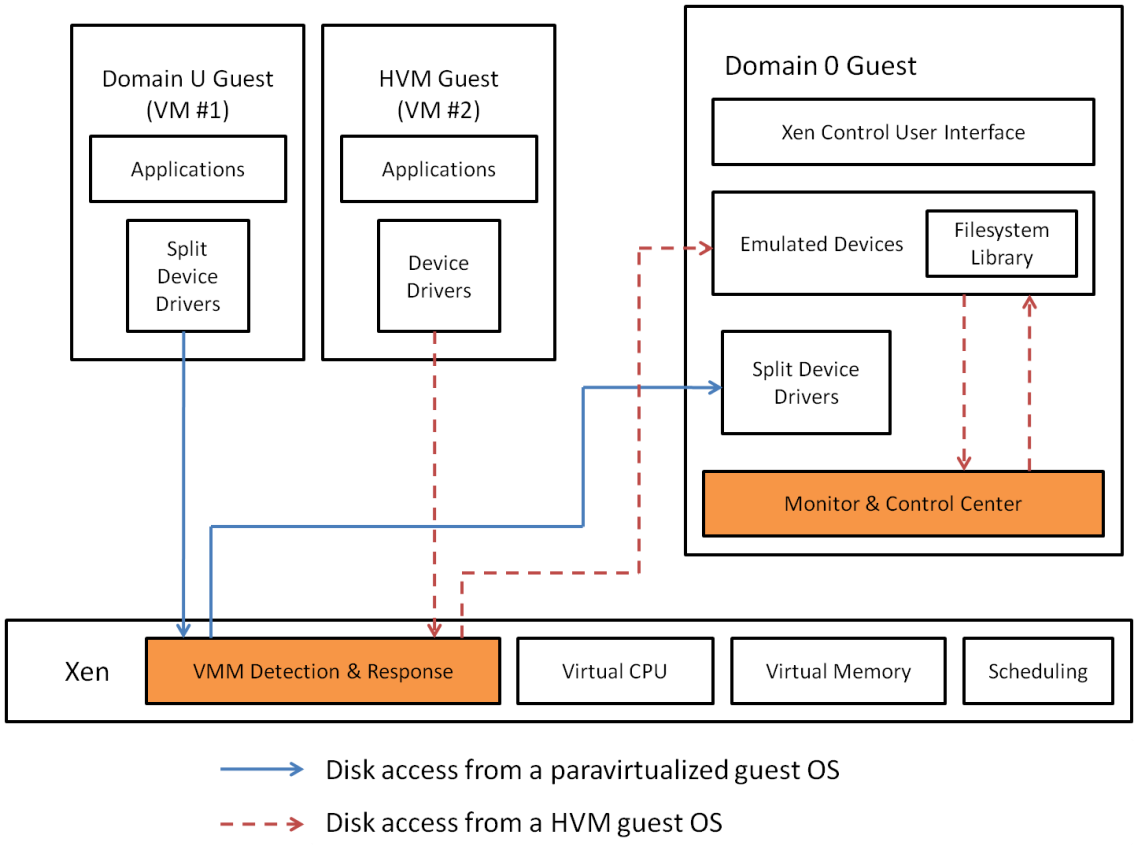
IDS/IPS 在某些情況下會需要對於磁碟上的檔案或磁區進行存取。比如說當防毒軟體進行全機掃描的時候，就需要能讀取磁碟機上的檔案、開機磁區來進行病毒偵測。或是使用如 Tripwire 之類的工具[101]對系統進行完整性(integrity)的檢測等均需仰賴檔案或磁區的存取。傳統上而言，這亦可透過 OS 的 API 來達到。比如說透過 ReadFile、DeviceIoControl 等 API。

我們所感興趣的是從 VMM 層直接存取 Guest VM 磁碟上的檔案或磁區。這雷同於透過 VMM 層直接存取 Guest VM 的記憶體。好處包括不需要安裝額外的模組至 Guest VM 內部、以及可以避免 Rootkit 遮蔽 API 的問題。

Xen 的虛擬磁碟有兩種格式。一個是以影像檔(Disk Image)的形式存放，一個影像檔對應到一個虛擬磁碟。另一個作法是把虛擬磁碟映射到一個實體的磁區或 Logic Volume (LVM)的作法。Disk Image 作法最明顯的好處是磁碟映象檔可以用 sparse file 的方式儲存，也就是說所配置的磁碟映象檔的實際大小可以隨實際使用量來擴增。而他的缺點是因為多了一層檔案存取的 abstraction，所以在存取速度上會較 LVM 的方式

來的慢。另外對於 online 的磁碟影像備份上無法利用現有的 LVM Snapshot 來協助維持備份過程中磁碟上資料的一致性(consistency)。

對於 Guest VM 之磁碟存取，Xen 本身已提供一些工具如 qemu-xen-nbd 可將磁碟映象檔模擬成系統上的裝置檔。再透過其他的系統工具如 kpartx 來辨識出該裝置檔的分割情況進而把分割區 mount 起來，這樣一來即可在 Domain 0 的環境下讀取 Guest VM 上的檔案達到 offline 的存取虛擬磁碟。



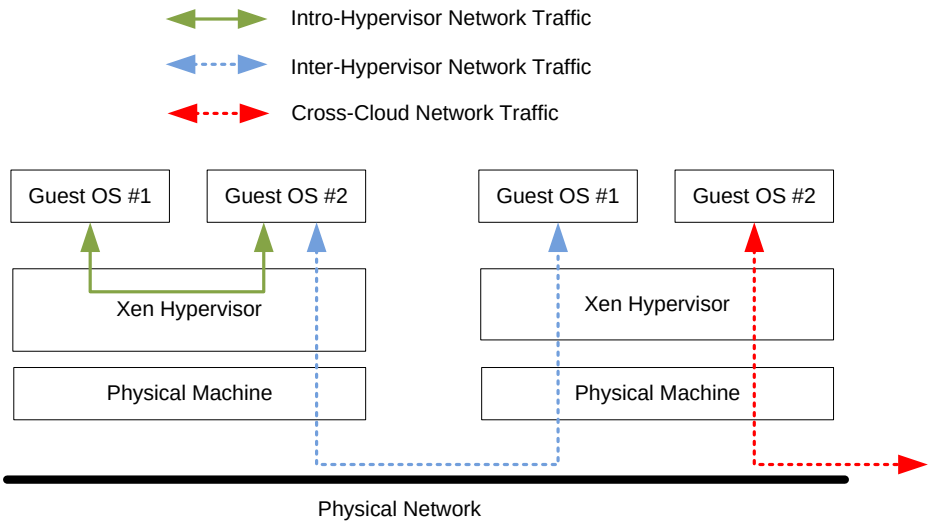
圖表十四：虛擬磁碟之監控

但是光有 offline 的虛擬磁碟存取是不夠的，在 online 形式的即時監控部份，目前已針對 HVM 形式之 Guest 有初步的成果，我們將 Qemu 與 debug-fs 及 ntfs-3g 的函式庫結合，讓 Qemu 不只看到 block level 也可以看得到 file level 的資訊，如此一來就可以在 Guest VM 運作的同時也能在 Domain 0 中動態地存取虛擬磁碟中的檔案，未來會再試著針對 paravirtualized 形式的 Guest VM 支援此功能。

當然另一方面，以 IPS 應用來說，很重要的一個功能是必需要提供對於 Guest VM 在存取磁碟動作上的控管。比如說當發覺某個檔案已感染病毒，必需要能夠阻擋對該檔案的存取。這方面我們可以從我們在 Xen Hypervisor 層中的 VMM Detection & Response 模組來達到控管效果。從上圖中我們可以清楚地看到，對於 Paravirtualize 的 Guest VM，他的磁碟存取是透過 split device driver 經由 Xen Hypervisor 然後轉至 Domain 0 上面另一半的 split device driver 來達成對實體磁碟(block device)的存取。另外如果是 HVM 形式的 Guest，磁碟存取則是經由 Xen Hypervisor 去存取位於 Domain 0 內部的模擬(emulated)的磁碟 driver (qemu-dm[90])來達成對實體磁碟的存取。也因此，在控管的

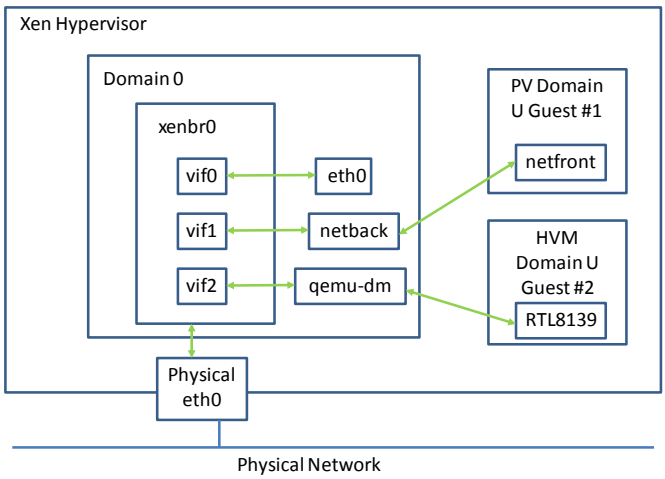
部分我們可在 Hypervisor 內部進行。但 Hypervisor 這一層級所看到的僅是對於各 block device 的各個單位(磁區)的存取要求。以實際使用上來說，我們必須要能夠將其對應到上層的檔案系統，如此的存取控管才有實際的意義。(可以試想一下要禁止存取一個受病毒感染的檔案，可是你又不清楚檔案是對應到哪個磁區的窘境)。因此我們配合我們所修改過的 qemu 來獲取關於磁區與檔案間的對應關係資訊而達到動態的監控與制動。

□ 網路觀測與制動技術

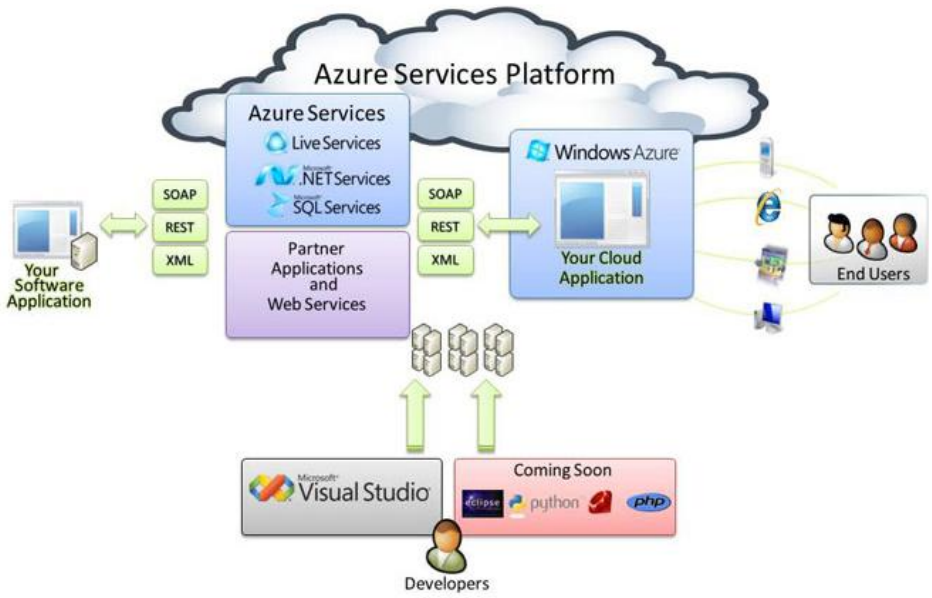


圖表十五：Network Traffic in a Xen-based IaaS Cloud

上圖提到的是 IaaS 雲環境歸類出來的種網路流動形式。第一種(Intro-Hypervisor Network Traffic)是屬於一台實體機器上之虛擬機器間的網路流量。次種流量並不需要透過對外的實體網路(Physical)，而是內部的網路流量。這樣的方式會對於一些特定傳統網路入侵偵測或是安全匝道這類的安全機制失效。原因出在網路入侵偵測模式或是安全匝道必須仰賴於實體網路的監聽和控管。對於這樣的機制就顯得束手無策。IaaS Cloud 對於這類的內部流量顯得尤為重要。



圖表十六：Virtual Network Interface in Xen



（圖片來源：Microsoft）

上圖是 Xen 虛擬網路的架構，對於 Paravirtualization (PV) Guest，其網路裝置的驅動程式是透過 split driver 的方式實現，而對於 HVM 形式的 Guest，網路則是透過 qemu-dm 去模擬硬體網卡（比如 Realtek 8139)來供 Guest OS。這樣的方式都會藉由虛擬網卡連到實體網卡，虛擬機就能夠透過虛擬網卡(vif0)對外聯絡。所以當虛擬機間要互相通訊的時候，就要透過虛擬網卡做溝通。

在這方面的話，既然要透過實體網卡，掌握 Dom 0 這部分所有的虛擬網卡和流量，就能全盤了解虛擬機之間的內部網路流量(Intro-Hypervisor Network Traffic)，但在這方面就必須修改 Dom 0 核心內部相對應網路配置的部分。可把問題改動成 Liunx Kernel 的重新編譯。只求觀測的情況下，那就是相對於處理 Linux 中 packet filter 所對應到的地方做出修正和功能性的延伸。原則上來說，流量經過 Dom 0 的時候都會被 Monitor 所攔截進行檢測和紀錄，軟體方面可以透過 libpcap 進行內部封包的抓取處理。

第二種 IaaS 流量是跨實體機器但仍在一朵雲內的網路流量（Inter-Hypervisor Network Traffic），這種流量就是跨越個別實體機器但仍屬於同朵雲內。而第三種是跨過雲邊界的網路流量（Cross-Cloud Network Traffic）也就是對雲端未知的外部進行連接。這兩種流量共通點就是都會經過實體網路(Physical Network)。這方面的對外網路流量，所有的 Guest 依樣都是透過虛擬網路卡連線到 Dom0 的實體網路，接著藉由實體網路卡連線到外部的網路，這樣的話所有的流量都會經過實體網路卡，只要能掌握 Dom 0 實體網路上面相對應的流量就能夠針對每個 Guest 的網路狀況達到監測和反制的作用。

- **基於雲端技術之網路安全實驗觀測平台**

  - **虛擬技術研究**

    現有的虛擬化技術透過不同的資源管理方式，讓使用者可以用比較好的方式（更低成本、更好的執行效率等）來執行所需的行程或執行緒。一般所指的虛擬化資源包括計算能力和資料儲存 [8]。虛擬化技術的呈現就是同時執行多個虛擬機（Virtual machine，簡稱 VM），透過軟體讓這些虛擬機（或為行程）像真實機器一樣執行其上的應用軟體。虛擬化技術又稱為平台虛擬化，目的是將作業系統和硬體平台切割。較常見的切割方式包括：

    - **完全虛擬化：**

      不需修改 Guest OS，所有軟體都能在虛擬機中執行，著名的例子有 IBM CP/CMS、VirtualBox、VMware Workstation 等。此類虛擬機技術執行效能較差（1.0GHz 的實體 CPU 僅能支援 <1.0GHz 的執行效能）。

    - **硬體輔助虛擬化：**

      不需要修改 Guest OS，利用 CPU 處理特殊的虛擬指令，以實現完全虛擬化的功能。著名的例子有 VMware Workstation、Xen、KVM 等。此類虛擬技術面臨的最大挑戰是硬體的支援度。如果使用太老舊的硬體，則無法透過實體 CPU 處理特殊的虛擬指令，並實現完全虛擬化的功能

    - **部分虛擬化：**

      只有部分應用程序以虛擬化方式執行。虛擬化處理未擴及整個作業系統。此類虛擬機無法支援多種 Guest OS 的運行。

    - Paravirtualization：

      需修改 Guest OS，為應用程序提供與底層硬體相似（但不相同）的操作介面。早期的 Xen 即採用此種虛擬切割方法。運行於此類虛擬技術上的虛擬機 Guest OS 受到較大的限制。

    綜合前面幾點所述，如果要同時兼顧虛擬機的執行效能（也就是 1.0GHz 的實體 CPU 可以執行 1.0GHz 的虛擬 CPU），並支持 Guest OS 的多樣性，則採用前述硬體輔助虛擬化的虛擬機技術將是較為可行的方法，然而，這種方法最大的挑戰之一就是軟硬體之間的相容問題。

    此外，近年來許多專家學者也指出虛擬機彼此之間可能潛在的一些安全議題、攻擊行為，也是在將虛擬機技術引入無線網路觀測平台時所必須特別注意的。在這個部分，研究人員也針對此一議題，深入探討，並將研究結果發表於 2012 年 IEEE SERE（International Conference on Software Security and Reliability）國際研討會中。

  - **測試平台研究**

    - Emulab

      Emulab, 由猶他大學所研發設計之網路測試平台，利用分散式系統與網路建置出一套研究用的仿真平台。在此平台中，使用者可以利用 NS 語法，將平台內的各實驗節點任意連接成一個網路拓樸。平台可以支援多組實驗同時進行，每個實驗網

路均以 VLAN 隔開。 不管實際接線情況為何，配置在同一個 VLAN 下的網路拓樸彼此之間可以互相通訊，就好似連接在同一個區網之內。此隔絕性可以保證各實驗之間彼此不會互相干擾。在 Emulab 中，使用者建置新實驗的步驟包括： 1) 配置實驗節點；2) 配置 VLAN，以建構所需的拓樸；3) 將映像檔載入所指定的實驗節點中。然後，使用者就可以開始進行其測試實驗。

- DETER

以 Emulab 為基礎，DETER 測試平台更進一步地提供安全保護機制，避免測試平台本身遭到外部攻擊，同時也保證測試平台內部的實驗資料（尤其是含有惡意程式碼的實驗）不會流入外部公用網路。此平台的特色在於提供「安全」的測試環境。其設計已經整合於目前的 Emulab 中，並有眾多使用者以此平台作為測試其新方法的依據。

- TESTBED@TWISC

Testbed@TWISC 由國內成功大學所建置之網路測試平台。此平台以猶他大學所授權的 Emulab 為基礎，具備網路測試所需的隔離性、仿真度等特性。由於國內外硬體規格差異，成功大學研發團隊在取得 Emulab 授權之後，便積極尋找可替代之國產硬體（包括實體實驗節點、電源控制器、交換器等設備），並自行研發相關軟、韌體，利用簡單的網頁操作介面，以因應國內資訊安全研究與實驗之測試所需。此測試平台中的實驗節點皆為實體機器，如前所述，使用者進行實驗之前，必須先配置得實驗節點。爾後，使用者便可以任意控制整個實驗節點（實體機器），包括 root 權限、作業系統與軟體安裝等。

- SWOON

SWOON 是國內交通大學與柏克萊大學專家學者共同開發研究之無線網路測試平台。以 DETER 為基礎，研發團隊設計各種不同的方法能仿真、模擬無線訊號，並實作 802.11 a/b/g、802.16d（WiMAX）等無線網路技術，能提供無線網路安全研究人員所需之實驗與觀測環境。此平台的設計挑戰在於無線訊號的飄移性與存取特性，因此，在訊號傳遞與隔離上採取以有線模擬無線的方式來實作。此外，802.11、802.16 及 802.15 等異質性的網路技術支援，也讓相關研究人員得以探索異質無線網路環境整合時所可能衍生的問題。此平台目前已實作一套簡易使用者操作介面，讓使用者可以透過 GTK+ 圖形化介面，簡易地配置其所需的無線網路拓樸，進行並觀察實驗結果。

比較上述兩個著重網路安全實驗的測試平台，DETER 與 SWOON，下表列出這兩個測試平台所能仿真之網路攻擊行為：

攻擊行為	C1	C2	C3	C4
War Driving	Not Emphasized	N	N	Y
MAC Spoofing		N	N	Y
IP Spoofing		Y	Y	Y
Eavesdropping		Y	Y	Y
Wireless Eavesdropping		N	N	Y
Man-in-the-Middle		Y	Y	Y
Evil Twin		N	N	Y
DDoS		Y	Y	Y

□ 虛擬機效能評估

確保虛擬機效能是vSWOON仿真度的一大挑戰。如果Boss伺服器配置一台1.0GHz的虛擬機給使用者，但是其執行效能卻只有 600MHz，則 vSWOON 的仿真度將因此受到波及。如此使用者所開發的系統，未來在實際環境運行時，可能會因實驗仿真度太低，而遭遇到無法預測的狀況。因此，本計畫執行期間，針對各種不同的虛擬技術運行效能進行評估，期望能提供 vSWOON 最佳的虛擬技術參考。實驗機器規格為 Intel Xeon　X3450 2.67GHz（8 核心中央處理器），搭載 10GB 的記憶體。計畫研究人員針對四種不同的虛擬機技術，以 lmbench 和 netperf 進行評測，包括 Xen、KVM、OpenVZ 與原生（Native）等機器，其中:

a. Xen 和 KVM 是使用自有工具的套件來管理 CPU 和 記憶體資源的使用。
b. OpenVZ 使用 cpu affinity 的功能來管理所使用的 CPU 數量。
c. Native system 使用 cpu affinity 綁定 cpu 資源, 並透過 PAM 管理能使用的記憶體大小。

貳、 計畫項目完成度

本計畫執行完成，總計畫及各子計畫已完成之具體項目如下：

- 已完成整合系統的整體架構設計與系統執行流程規劃。
- 已完成保護隱私雲端入侵偵測子系統之設計與雛型系統之實做。
- 已設計出一個可抵擋 chosen plaintext attack 之公開金鑰加密系統。
- 已完成非集中式雲端儲存系統修復機制之背景研究，並已針對現行的技術提出改良方案。
- 已完成使用者端資料加密容錯編碼之雛型系統之實作。
- 已提出一項可授權驗證的資料完整性檢查方法。
- 已完成網路檔案蒐集子系統之設計與雛型系統之實作。
- 已完成分散式資料儲存子系統之設計與雛型系統之實作。
- 已完成文件 Call/Pop 分析子系統之設計與雛型系統之實作。
- 已完成惡意程式分析子系統之設計與雛型系統之實作。
- 已完成全系統層次的行為分析系統之設計與雛型系統之實作。
- 已完成 Hypervisor 層系統呼叫攔截子系統之設計與雛型系統之實作。
- 已完成記憶體內容物件的讀取、解析與制動子系統之設計與雛型系統之實作。

以下將上述完成項目與本計畫所規劃之工作完成度，依總計畫以及子計畫分別列出說明：

- **總計畫工作項目完成度:（完成度：100%）**

  總計畫執行完成並符合進度規劃，其具體工作項目及進度如下：

  - **規劃整體計畫研究架構（完成度：100%）**

    總計畫已規劃完成整合系統的完整架構，特別是與子系統間溝通介面的部份以及功能性的畫分。
  - **建立各子計畫經驗交流機制（完成度：100%）**

    總計畫已安排常態性的進度會議與經驗交流活動，可有效避免各子計畫在功能設計上的衝突，研發人員之間亦可收到截長補短的效果。
  - **設計整合系統之基礎軟硬體架構（完成度：100%）**

    總計畫已完成針對各子計畫所使用的軟硬體設備進行調查與統合。
  - **技術成果整合（完成度：100%）**

    各子計畫已有成果產出，總計畫已完成各項技術成果與整體系統架構的比對與整合，並確認各子計畫的技術產出與原規畫之進度相符。
  - **撰寫成果報告（完成度：100%）**

    各子計畫已回報各項研究成果與技術細節資料，總計畫完成成果整理與報告撰寫工作。

- **支援多樣功能之雲端資料安全儲存（完成度 100%）**

本子計畫之工作項目可分為保護隱私的雲端入侵偵測、非集中式雲端儲存系統中系統修復機制、資料完整性授權驗證機制等項目，以下將各別說明其完成度：

- **保護隱私的雲端入侵偵測（完成度 100%）**

雛型系統：保護隱私的雲端入侵偵測系統

說明：在此項目中我們實作了一個能保護使用者上傳資訊的隱私性且又能夠提供入侵偵測的雲端入侵偵測系統。

- **非集中式雲端儲存系統中系統修復機制（完成度 100%）**

論文產出：Hsiao-Ying Lin, Wen-Guey Tzeng, and Bao-Shuh Lin. A Decentralized Repair Mechanism for Decentralized Erasure Code based Storage Systems. IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2011.[34]

說明：在此項目中為 (n,k) 分散式容錯編碼分散式儲存系統設計了一個具有更好效率的修復機制，且該機制能一次修復多個錯誤。此項目的研究的成果已經獲得國際學者的肯定[34]，同時能夠進一步完善我們先前建構的安全非集中式儲存系統。

雛型系統：使用者端資料加密容錯編碼之處理系統

說明：我們實作了一個在使用者端執行的資料加密容錯編碼處裡系統，利用加密技術保護資料的隱私性，並且利用容錯編碼保護資料的可靠性。

- **資料完整性授權驗證機制（完成度 100%）**

論文產出：Shiuan-Tzuo Shen and Wen-Guey Tzeng. Delegable Provable Data Possession for Remote Data in the Clouds.(Accepted by ICICS 2011)

說明：在此項目中我們提出了一個資料完整性驗證方法，其能夠達到

（1）有效率的資料完整性驗證

（2）驗證能力的授權

（3）被授權人再授權的防止

（4）驗證能力的註銷

而其結果已被國際研討會 ICICS 2011（Thirteen international conference on information and communications security）發表。我們的研究成果使得資料完整性驗證有更實際的應用情境，解決了在資料分享的情境下，隱私的資料驗證方法，以及更廣泛更多樣化的應用可能，同時也開啟了嶄新的研究方向。

- **雲端惡意程式分析平台（完成度：100%）**

本子計畫之工作項目可分為網路檔案蒐集、分散式的資料儲存、惡意程式分析、文件 Call/Pop 分析器等項目，以下將各別說明其完成度：

- **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，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。

- **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。

- **惡意程式分析(完成度：100%)**

  本子系統分析程式模組化，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，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。

- **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，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，我們已結合利用 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。

- **全系統層次的行為分析系統(完成度：100%)**

  本分析系統採用動態污染分析技術(Tainting Analysis)，乃透過追蹤電腦運作時程式與資料間的相互影響關係，以找出程式的異常行為。每當 CPU 執行一行指令後，某些變數便會受到其他某些變數的影響，使其數值發生變化，這些變化常代表某些重要事件的發生。本系統運用一個汙染源分析引擎，來追蹤一個惡意程式樣本在本系統中所造成的資訊流動。透過整合工作，本系統提供一個微軟視窗作業系統為基礎的作業系統環境，來檢驗該程式。而在此，本系預先設定好汙染源（Taint Source）和汙染目標（Taint Sink），自動地分析雲端惡意程式分析平台上所蒐集到的樣本。此系統會蒐集汙染源分析引擎所標記的所有二進位檔案。而這些二進位檔案即是因目標惡意程式對系統所產生的影響。將上面步驟所蒐集

到的二進位檔案，還原至具有系統意義的物件。當今的電腦系統，所有的物件都可以用二進位來表示，例如檔案、程序等等。而這些二進位檔案對於惡意程式行為分析不具有意義。我們需要把這些二進位檔案，還原至分析人員比較注重的系統資訊，例如程序名稱、檔案內容、運行程式清單、載入驅動程式等等。這些系統層次物件，才具有實質的意義。接下來為惡意程式行為解析，透過蒐集到所有被影響的系統物件，我們可以總結該惡意程式的行為。舉例來說，若一個惡意程式影響到了一些登錄檔的鍵值，我們可以總結「該樣本具有異動登錄檔的行為」。相同道理，可以針對程序的創建，驅動程式、檔案系統、開機磁區、網路通訊等等。來全面性的觀測一個惡意程式樣本。本分析系統不但加強了第一年度所開發的分析平台的分析能力，其中一些衍伸的技術以及累積的分析經驗，在實務上有著明顯的幫助。

- **基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制(完成度 100%)**
  - **從 Hypervisor 層攔截系統呼叫(完成度 100%)**

    現在的許多惡意程式常常會產生特定的系統呼叫組合或是產生頻率異常多的系統呼叫，所以我們能夠從監控 Guest VM 的系統呼叫來判斷是否可能有惡意程式的執行。本分析模組從 Dom 0 的 Monitor and Control Center 經由我們所設計的 Hypercall 直接對 Xen Hypervisor 發出需求，利用讀取被監控的 Guest VM 的暫存器的值來找到負責處理系統呼叫函式的記憶體位置，並利用插入特權指令造成 VMExit 的方法來攔截系統呼叫。
  - **記憶體內容物件的讀取、解析與制動 (完成度 100%)**

    以系統的角度來說，記憶體是一項不可或缺的資源，任何等待被執行的 process 都需要經過 scheduling 拿到 cpu cycle 以及合法的 memory space 才可在 OS 內運行，也正因為如此，以記憶體為基礎的防護機制就變的相當重要，不管是一個等待被執行的惡意程式、著名的 buffer overflow 或是一段被插入在 memory space 的 nop-slay malicious code，都可以藉由記憶體內容物件的讀取、解析與制動來加以防範。本分析模組在 Monitor and Control Center 經由 Xen Hypervisor 所提供的 API (xc_map_foreign_range) 來達到讀取被監控對象(Guest VM)的記憶體內容。但是從記憶體拿出來的內容是屬於 low level information，我們得透過解析將這類資訊轉換成 high level information，並從中取得 process 的詳細資訊。轉換成 high level information 後，目前我們仍在嘗試如何解析記憶體中可疑指令和程式流程，得到解析結果之後，我們便可進一步利用 Intel VMX(Virtual Machine Extensions)所提供的 EPT(Extended Page Table)對 process 所屬記憶體進行存取權限管理，透過修改 EPT 裡 memory block 的 access bits(Read、Write、Execute)來管理該 memory block 是否能被 process 使用，進而達到反制惡意程式的效果。
  - **開發 online 式的磁碟檔案系統及磁區觀測及解析、制動技術(完成度 100%)**

    很多時候，當駭客入侵系統時，有可能只是先將後門程式植入(存放)在被入侵的系統的磁碟上，他們並不會立刻展開全面性的攻擊或對系統內部核心的入侵，而是等到一個不引人注目的時間，才透過植入的後門或木馬程式悄悄發起攻擊，此攻

擊很可能會造成異常的磁區讀寫情況，我們的系統透過磁區讀寫的觀測、記錄並解析其是否有異常狀況發生，進而能夠對惡意的攻擊行為進行阻擋。

在磁區觀測的部份，由於 Xen Hypervisor 對 Guest VM 的週邊裝置大多是藉由 Qemu 模擬，特別是 HVM 形式的 Guest VM，因此所有的磁區操作最後都會經由 Qemu 來讀寫該系統的區塊裝置(像是映像檔或者是硬碟上的某一個分割區)。我們藉由修改 Xen 中 Qemu 的程式碼(位於 tools/ioemu-qemu-xen/，主要是 block.c 及 aio.c 這兩個檔案)，對負責磁區讀取和寫入函式的呼叫做監控來達到我們所想要的目的，在 Qemu 收到 Hypervisor 所轉換的磁區存取請求後，我們即可在函式中得知其磁區存取的位置，每一次磁區寫入的位置都會被記錄下來。

而關於制動的部分，我們在偵測到系統受攻擊時，可即時地在 Qemu 中將所有磁區存取做過濾的動作，這樣便能將惡意的存取動作擋下來，而過濾的機制則透過我們修改過結合 debug-fs 及 ntfs-3g 函式庫的 Qemu，得知關於檔案及使用者權限等其他有關檔案系統的資訊，以及需受保護的檔案實際上是存在磁區的哪些位置，整合至 Dom0 的控制中心後就能在過濾時判斷哪些存取請求該擋下來哪些則不該擋，我們也可以主動透過 Qemu 把可疑檔案內容讀取出來，並掃描這些檔案的內容，根據掃描的結果決定是否透過 Qemu 將該檔案刪除。

- **網路觀測與制動技術(完成度 100%)**

  透過 Xen Hypervisor 對 Guest VM 進行網路封包系統防護與惡意流量監測，主要著手於攔截 Guest VM 的封包、監測 Guest VM 封包的流量，以及針對 Guest VM 的網路封包系統進行 online 式的觀測與解析，並將 Snort 整合進 Xen Hypervisor 大架構中。由於 Guest VM 的網路封包在對外或對內的流通時，最終都需要經過 Dom 0，所以我們將擷取 Guest VM 的封包，接這對於不同目的地不同目標的封包都進行監控與防護，使用適切的 snort rules 達到防護與效能堅固的目的。

- **基於雲端技術之安全實驗觀測網路(完成度 100%)**
  - **研究現有雲端技術與雲端服務(完成度 100%)**

    現有雲端技術與雲端服務之研究，請參考本文壹-四-基於雲端技術之網路安全實驗觀測平台。
  - **研究現有網路測試與觀測技術(完成度 100%)**

    現有網路測試與觀測技術之研究，請參考本文壹-四-基於雲端技術之網路安全實驗觀測平台。
  - **結合雲端技術與網路觀測，設計新的雲端安全實驗觀測網路(完成度 100%)**

    結合雲端技術與網路觀測，設計新的雲端安全實驗觀測網路。請參考本文請參考本文壹-四-基於雲端技術之網路安全實驗觀測平台-vSWOON 說明
  - **利用虛擬機技術，設計並實作雲端安全實驗觀測網路之管理機制(完成度 100%)**

    利用虛擬機技術，設計並實作建置雲端安全實驗觀測網路之管理機制。使用者不需要強大運算功能的裝置或設備，將複雜的運算工作或實驗交給雲端處理。
    - 請參考本文第三節 E 項說明。
    - 本計劃建置虛擬機效能評測實驗，實驗結果可提供雲端安全實驗觀測網路之建置參考，以及其資源管理機制的參考依據。

參、 計畫成果

本計畫執行結束所獲得之具體成果可分為產學合作計畫、學術貢獻、系統建置等部分，統計如下：

產學合作計畫	兩年度合計已簽訂 12 項產學合作計畫 計畫金額達 16,863,500 元
國內外專利	兩年度合計已申請通過一項專利，另有兩項專利申請中
期刊論文	兩年度合計已發表 9 篇國際期刊論文
會議論文	兩年度合計已發表 16 篇國內與國際會議論文
雛型系統研發	兩年度合計已研發完成共 9 個子系統之雛型系統

圖表十七：計畫成果一覽表

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，計畫團隊已陸續與國內業界大廠及政府研究機構簽訂產學合作計畫，其計畫名稱與合約金額如下表二：

合作計畫名稱	合作對象	合約金額
中華電信研究所委託研究計畫案-行動平台資通訊安全問題的研究(三年期合計)	中華電信	2,880,000
動態惡意程式行為側錄與污染分析	中華電信	905,000
DNSsec 推動先期型計畫(二年期合計)	教育部	4,000,000
委託 Open D-Link Routers Forum 建置、維護與測試技術與諮詢服務(II)	友訊科技	1,500,000
惡意程式自動檢測技術支援系統委託研究案	法務部調查局	784,000
前瞻性檔案完整性驗證與可疑嵌入碼檢測平台	喬鼎科技	950,000
Android 應用程式隱私資訊竊取針測系統	工業技術研究院	600,000
雲端惡意程式鑑識與行動平台安全	宏達電子(HTC)	2,000,000
雙階層式全系統汙染鑑識分析	中華電信	945,000
Technology Transfer on Network Threat Detection using Security Log Correlation	趨勢科技	1,200,000

虛擬機效能瓶頸之非侵入式偵測	工業技術研究院	499,500
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	600,000

圖表十八：產學合作計畫表

二、 學術貢獻

□ 期刊論文:

(1) Yu-Lung Huang, Chiya Shen, Shiuhpyng Shieh, "S-AKA: A Provable and Secure Authentication Key Agreement Protocol for UMTS Networks," accepted for publication, *IEEE Transactions on Vehicular Technology*.

(2) H.-Y. Lin, W.-G. Tzeng."A Secure Decentralized Erasure Code for Networked Storage Systems," *IEEE Transactions on Parallel and Distributed Systems*, 21(11), pp.1586-1596, 2010.

(3) T. Kløve, T.-T. Lin, S.-C. Tsai, W.-G. Tzeng."Permutation Arrays Under the Chebyshev Distance," *IEEE Transactions on Information Theory* 56(6), pp.2611-2617, 2010.

(4) C.-L. Hou, C. Lu, S.-C. Tsai, W.-G. Tzeng."An Optimal Data Hiding Scheme with Tree-Based Parity Check," Accepted by *IEEE Transactions on Image Processing*, 2010.

(5) Chia-Wei Hsu, C.W. Wang, Shiuhpyng Shieh, "Reliability and Security of Large Scale Data Storage in Cloud Computing," *IEEE ATR*, 2011.

(6) Pokai Chen, Shiuhpyng Shieh, "Security for Future Internet Architecture - Motivation from DNSSEC," *IEEE ATR*, 2011.

(7) Hsiao-Ying Lin, Wen-Guey Tzeng."A Secure Erasure Code-based Cloud Storage System with Secure Data Forwarding," *IEEE Transactions on Parallel and Distributed Systems* 23(6). pp.995-1003, 2012.

(8) LY, Yeh, Y.L. Huang, S.P. Shieh, Anthony Joseph, "A Batch Authenticated and Key Agreement Framework for P2P-based Online Social Networks, " accepted for publication, *IEEE Transactions on Vehicular Technology*.

(9) Hsin-Yi Tsai; Siebenhaar, M.; Miede, A.; Yulun Huang; Steinmetz, R. , "Threat as a Service?: Virtualization's Impact on Cloud Security," *IT Professional*, vol.14, no.1, pp.32-37, Jan.-Feb. 2012

□ 會議論文:

(1) Yi-Ruei Chen, J.D. Tygar, W.-G. Tzeng, "Secure Group Key Management Using Uni-Directional Proxy Re-Encryption Schemes," *In the 30th IEEE International Conference on Computer Communications (INFOCOM 2011)* pages 1322-1330, 2011.

(2) Hsiao-Ying Lin, Wen-Guey Tzeng, and Bao-Shuh Lin."A Decentralized Repair Mechanism for Decentralized Erasure Code based Storage Systems." *IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2011.

(3) Shiuan-Tzuo Shen and Wen-Guey Tzeng."Delegable Provable Data Possession for Remote Data in the Clouds." Accepted by *International Conference on Information and Communications Security* (*ICICS)*, 2011

(4) Bing-Han Li, Shiuhpyng Shieh, "RELEASE: Generating Exploits Using Loop-Aware Concolic Execution," *IEEE Conference on Secure Software Integration and Reliability Improvement*, June 2011

(5) 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.

(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.

(7) Mung-Lu Tsai, Yen-Chun Hsu and Yu-Sung Wu, "An IaaS Cloud for Attack and Defense Experiments," *21th Cryptology and Information Security Conference (CISC)*, 2011.

(8) Hsiao-Ying Lin, Shiuan-Tuzo Shen, Wen-Guey Tzeng, Bao-Shuh Lin."Toward Data Confidentiality via Integrating Hybrid Encryption Schemes and HDFS." In the *26th IEEE International Conference on Advanced Information Networking, (IEEE AINA 2012)*, March, 2012

(9) Hsiao-Ying Lin, Wen-Guey Tzeng, Shiuan-Tzuo Shen and Bao-Shuh P. Lin. A Practical Smart Metering System Supporting Privacy Preserving Billing and Load Monitoring." In the*10th International Conference on Applied Cryptography and Network Security (ACNS 2012)*, June 2012.

(10) Hsiao-Ying Lin, John Kubiatowicz and Wen-Guey Tzeng."A Secure Fine-Grained Access Control Mechanism for Networked Storage System." *In the Sixth IEEE International Conference on Software Security and Reliability (IEEE SERE 2012)*, June 2012.

(11) Yi-Ruei Chen, Wen-Guey Tzeng."Efficient and Provably-Secure Group Key Management Schemes Using Key Derivation." *In the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-2012)*, June 2012.

(12) Kuei-Yi Chou, Yi-Ruei Chen, Wen-Guey Tzeng, "An Efficient and Secure Group Key Management Scheme Supporting Frequent Key Updates on Pay-TV systems," In the*13th Asia-Pacific Network Operations and Management Symposium (APNOMS 2011)*, September, 2011.

(13) Chia-Ming Fan, Shiuhpyng Shieh, Bing-Han Li, "On the security of password-based pairing protocol in Bluetooth." *APNOMS 2011*: 1-4

(14) Yun-Min Cheng, Bing-Han Li, Shiuhpyng Winston Shieh, "Accelerating Taint-Based Concolic Testing by Pruning Pointer Overtaint." *SERE 2012*: 187-196

(15) Yu-Lun Huang, Borting Chen, Ming-Wei Shih, Chien-Yu Lai, "Security Impacts of Virtualization on a Network Testbed." *SERE 2012*: 71-77
(16) 陳仲寬、陳威志、許家維、謝續平，「潛在惡意程式找尋與行為分析之科技犯罪調查」，*第二十二屆全國資訊安全會議*

□ 專利

(1) S.I. Huang, S.P. Shieh,"Method and System for Secure Data Aggregation in Wireless Sensor Networks," China patent number ZL200710301500.9, 2011.
(2) (申請中) 王繼偉、謝續平、張佳惠，「以分離式的資訊流動追蹤偵測 Android 應用程式隱私竊取之方法」
(3) (申請中) 許家維、李秉翰、謝續平，「具虛擬機器感知能力程式之偵測方法以及系統」

三、 系統建置

本計畫所建構出之各子系統簡述如下：

- **支援多樣功能之雲端資料安全儲存**

  此部份之研究主題包含以下三項成果：

  - **保護隱私的雲端入侵偵測**

  (1) 建立雲端環境

  本實驗中我們採用三台筆記型電腦當作我們的雲，每一台電腦是雙核心 1.4GHz，記憶體 2GB，硬碟 320GB，作業系統是 Ubuntu。安裝 Hadoop 來建立雲的環境，其中一台電腦為 Master，另外兩台為 Slaver。

  在安裝 Hadoop 之前，要先將每一台電腦的網路給設定好。這個網路設定是為了確保每一台電腦可以正確的通訊，在本實驗中，我們將三台電腦分別設定為 192.168.0.161, 192.168.0.162, 192.168.0.164。可以利用"ping"或"telnet"來測試每一台電腦是否可以正確的通聯。因為 Hadoop 必須執行在 Java 虛擬機器上且版本必須為第 6 版以上，所以我們要安裝 Sun Java 6 的套件，可用"apt-get install sun-java6-jre"來完成 Java 的安裝。安裝完 Java 後，我們必須確認 Java 安裝的路徑，在本實驗中為 "/usr/lib/jvm/java-6-sun"，這個路徑是要用來設定 Hadoop 的設定檔，以確保 Hadoop 可以正確地找到 Java 安裝的位置。因為 Hadoop 是採用 SSH 當作每一個節點彼此的通訊方式，因此我們安裝 OpenSSH 以及設定每一個節點用來認證彼此的金鑰。安裝及設定完後可用"ssh ip"來確認安裝及設定是否成功。

  完成好上述準備工作後(網路設定→安裝 Java→安裝設定 SSH)，接著我們才可以開始安裝 Hadoop。Hadoop 的安裝只要注意到其安裝的位置和使用者權限即可，在本實驗中我們的安裝路徑為"/opt/hadoop-0.20.2"(其版本為 0.20.2)，使用者為"hadoop"。安裝為後要設定"hadoop-env.sh"、"core-site.xml"、"hdfs-site.xml"、"mapred-site.xml"、"master"、"slaver"等相關檔案。完成其相關設定後我們可以開啟瀏覽器輸入：http://node1:50030 來檢查 Hadoop 是否可以運行。其運行圖如下圖所示。

Cluster Summary (Heap Size is 45.31 MB/888.94 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	0	2	4	4	4.00	0

NameNode 'node1:9000'

Started: Sun May 22 20:27:52 CST 2011
Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

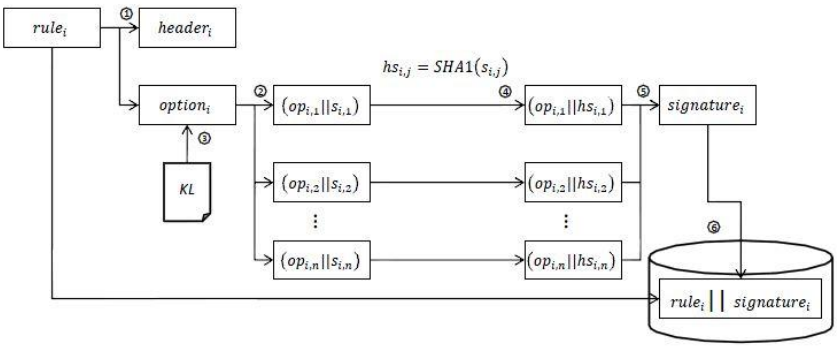
9 files and directories, 2 blocks = 11 total. Heap Size is 45.31 MB / 888.94 MB (5%)

Configured Capacity	:	70.56 GB
DFS Used	:	36 KB
Non DFS Used	:	7.15 GB
DFS Remaining	:	63.41 GB
DFS Used%	:	0 %
DFS Remaining%	:	89.87 %
Live Nodes	:	1
Dead Nodes	:	0

圖表十九：Hadoop 執行狀態

(2) 建立關鍵字資料庫

我們的入侵偵測服務是對電腦的網路封包做關鍵字比對，比對封包是否有和事先定義好的攻擊特徵符合。一個封包可以分為 header 和 payload 兩大部分，我們針對這兩部分做分析和比對，來檢查每一個封包內是否含有惡意的特徵值。因此，在 SNORT rule 當中，我們針對 Payload Detection 和 Non-payload Detection 兩大部分做分析和研究，希望可以找到最少組的關鍵字來做比對又可以涵蓋大部分的 SNORT rule。分析完 SNORT 規則後，我們選出三個關鍵字："content"、"flow"、"icode"當作關鍵字來轉換 SNORT 規則來成為我們的關鍵字資料庫。轉換的方法是保留上述所提到的關鍵字，加上述關鍵字的內容經過 SHA-1 固定長度後附加在原來規則的後方當作比對的關鍵字資料。最後我們將轉換後的關鍵字資料庫放到雲端上儲存。其轉換方法如下圖所示。



圖表二十：SNORT rule 轉換方法和過程

SNORT rule 中含有大約 8800 條左右的規則，在 Figure 3 中我們表示為 ，而每一個 可分為 和 兩部分，其中 包括 SNORT rule 的行為、來源方 IP 和 PORT、接收方 IP 和 PORT、網路流向等部分，而 是 SNORT rule 用來做比對的內容，每一個 又分為 和 ，其中 代表 中要比對的關鍵字，而 帶關鍵字裡面要比對的內容。舉例來說，假設 是 content:hello; ttl:64;，則 =content， =hello，其比對方式是比對每一個封包內的 payload 部分，看是否含有 hello 這個內容，而 =ttl， =64，其比對方式是比對每

一個封包內 ttl 這個 header 是否其內容為 64。而比對的部份我們只關心 Payload 和 Non-payload 的部分，所以我們將我們要比對的關鍵字做成 Keyword List(KL)來過濾分析 的內容。將取出的 中的 的部分我們使用 SHA-1 雜湊函數來固定其長度和格式，表示為 ，然後將原先的 =‖ 轉換為 ‖ 。將每一個收集起來成為對應 的 ，然後將 ‖ 放到雲端上儲存，當作是先定義好的特徵值。

(3) 保護隱私方法

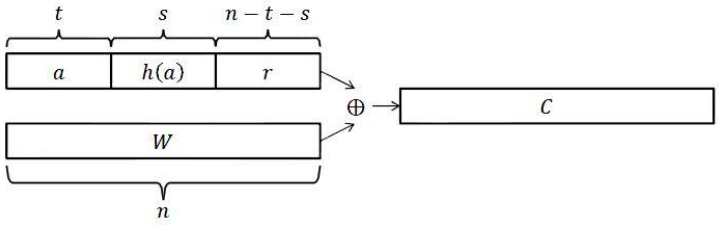
我們利用 Song, Wanger, Perrig 的方法和 Java 撰寫出對應的 client-server 程式來達到保護使用者上傳資訊的隱私。這個程式分為兩個部分：client 和 server。

Client 這程式具有以下能力：收集使用者電腦的封包、分析和轉換收集來的封包、加密收集來的封包、儲存收集來的封包、上傳資訊到雲端、比對候選者名單和儲存收集過的封包來判斷是否遭受惡意攻擊。以下針對上述功能做一個簡單的介紹(以下含有程式碼的地方，其內容和參數為本實驗所設定的值)。

Client 端功能
● 收集使用者電腦的封包 我們使用"jpcap"這個 library 來完成收集封包的功能。首先找到相對應的網卡" JpcapCaptor captor = JpcapCaptor.openDevice(devices[Integer.parseInt("4")], 2048, false , 10); "，然後開始收集流經這張網卡的封包" captor.loopPacket(-1, ps); "。
● 分析和轉換收集來的封包 由於我們是採用 SNORT rule，而 SNORT 目前只支援 ip, tcp, udp, icmp 等協定，所以我們針對這幾個協定收集及封包" p = ps.getPacket(); "，" if(p != null && p instanceof IPPacket) "，" if(p != null && p instanceof TCPPacket) "，" if(p != null && p instanceof UDPPacket) "，" if(p != null && p instanceof ICMPPacket) "，上述功能就可將封包 header 的部份給收集起來；因為 SNORT 並不是每一個 header 都有比對，所以我們根據我們分析 SNORT 的結果，將 SNORT rule 有可能比對得 header 給找出來，" tcp.sequence "，" icmp.code "，etc..。另一方面，由於 SNORT rule 有很大的部分都是比較 payload 的部分，所以我們還必須收集封包內 payload 的部分" byte[] PData = p.data; "，將收集來的 payload 轉換成 16 進位的 ASCII 的格式表示。
● 加密收集來的封包 為了保護上傳到雲端上的封包的隱私，我們對這些封包做一個加密的動作。一般而言，一個封包裡面會含有許多的 header，而我們只對 header 的內容做加密，然後將 header 轉換成我們是先定義好的關鍵字。舉例來說，假設我們先前收集到的封包含有 header "icode"且其內容是"15"，則我們後將"15"做加密並且保留"icode"這個關鍵字；假設我們先前收集到的封包的 payload 為"47 48 49"(16 進位 ASCII 表示法)，則我們

對"47 48 49"做加密並對其附加一個關鍵字叫做"content"。

接著我們要說明我們加密的方法。由於收到要加密的內容長度都不固定，所以我們會利用 SHA-1 雜奏函數來將其長度固定。加密的方法如下圖所示。首先加密的內容經過 SHA-1 固定長度為 W，為了加密 W 我們要產生一把 key=a||h(a)||r 來和 W 做 xor 這個運算。Key 產生的方式為先隨機選擇一個 t bits 長的隨機數 a，接著用 hash 算出 h(a)的值並且取其前面 s bits，本實驗用的 hash 是 SHA-1 這個雜奏函數，最後再取一個 n-s-t bits 長的隨機數 r(n 是 W 的 bit 數)，則 key 就如下圖所示：key = a||h(a)||r。



圖：加密演算法示意圖

- 儲存收集來的封包

我們要將收集來的封包暫時儲存在 client 端，以等待 server 端傳回的候選者名單做最後的比較。在這裡我們是將每一個收集到的封包用"時間序號(從 1970 年 1 月 1 日 00:00:00 開始算)"當做 index，用"map"這個結構的方式儲存在 client 端。

- 上傳資訊到雲端

利用 socket programming 的方式" out.writeObject(sendlist); "，" out.flush(); "，將前面加密過的封包上傳到遠端的雲上。

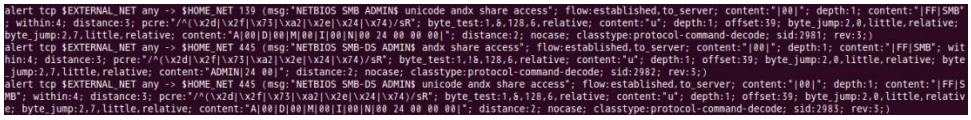
- 比對候選者名單和儲存收集過的封包來判斷是否遭受惡意攻擊

將雲傳回來的候選者名單和儲存的收集來的封包直接做明文比對，來判斷之前上傳到雲端的封包是否為惡意攻擊。

目前這個程式只支援在一般電腦上執行。使用該電腦的使用者必須擁有 root 以上的執行能力(為了打開網卡收集封包)且該電腦必須安裝 jpcap 這個 library(支援收集封包的 library，該 library 目前不支援 64bit 的作業系統)和 Sun-Java(執行 Java 程式)。接著在一般電腦上執行"sudo java client"，即可開始執行上述功能。執行畫面如下圖所示。

圖表二十一：收集封包示意圖

圖表二十二：封包加密上傳示意圖



圖表二十三：收到候選者名單示意圖

Server 這程式具有以下能力：讀取和儲存關鍵資料庫、做關鍵字比對來分析上傳的封包是否為可能的惡意攻擊、回傳候選者名單給使用者做最後的判斷。

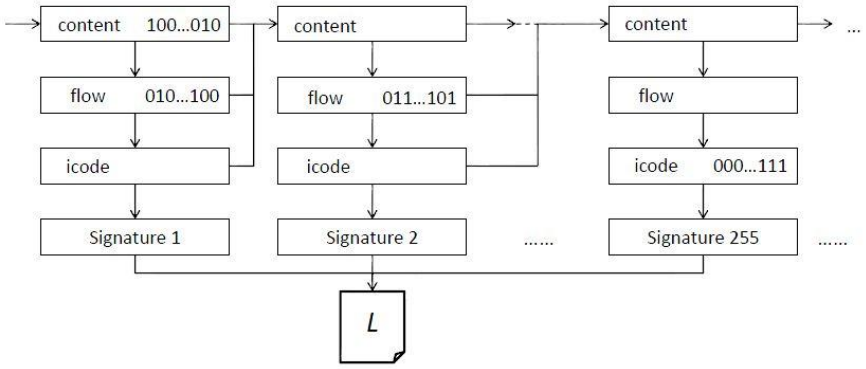
Server 在收到 client 的 request 時，Server 會先將收到的 request 做一個前置處理，這個處理主要是要取出從 client 端傳來封包的時間序號，我們將這個時間序號當作 index 來識別每一個上傳的封包；接著我們將其他上傳的資訊寫入一個檔案裡面，用這個 index 當作檔名，上傳到 Hadoop 的 HDFS 上面" Process p =

run.exec("/opt/hadoop-0.20.2/bin/hadoop dfs -moveFromLocal " + name + " ."); "；最後我們在 Hadoop 上寫一個 www 的程式來做關鍵字比對的功能，當 Server 收到 client 的 request 時，就會去執行 www 這個程式來完成關鍵字比對的工作。

www 這個程式的功能是讀取和儲存關鍵資料庫以及做關鍵字比對且 使用 Hadoop 自動地將工作分配給 slaver 這些 nodes 去執行。首先 www 要先找到儲存在 HDFS 上的關鍵字資料庫"　FileInputFormat.setInputPaths(conf, new Path("/user/hadoop/sig"));　"，接著取出其內容" map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) "，" String line = value.toString(); "，另一方面，我們還必須將前面所提到的 index 這個檔案從 HDFS 中找出來" patternsFiles = DistributedCache.getLocalCacheFiles(job); "，

" BufferedReader fis =

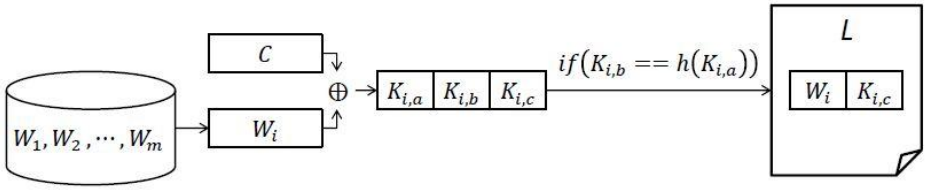
new BufferedReader(new FileReader(patternsFile.toString())); "；找出來之後我們利用下面流程來和關鍵字資料庫的資料做比對如下圖所示。



圖表二十四：關鍵字比對流程

我們將從關鍵字資料庫取出來的資料稱作 datai = rulei||signaturei，而 index 這個檔案裡個資料稱作 packet。其中 signaturei 要和 packet 做比對，當比對正確時，我們才會將 rulei 放到候選者名單 L 中，當作回答傳回給使用者。每一個 signaturei 和 packet 都會含有"content"，"flow"，"icode"這三個關鍵字，比較的內容是關鍵字後面的那一串二進位的值，其比對的流程是先比"content"，若正確則往下比"flow"，若依然正確則繼續往下比"icode"，反之一旦失敗，則比下一條 datai。若關鍵字後面沒有那一串二進位的值，則當作正確，繼續比下一個關鍵字。

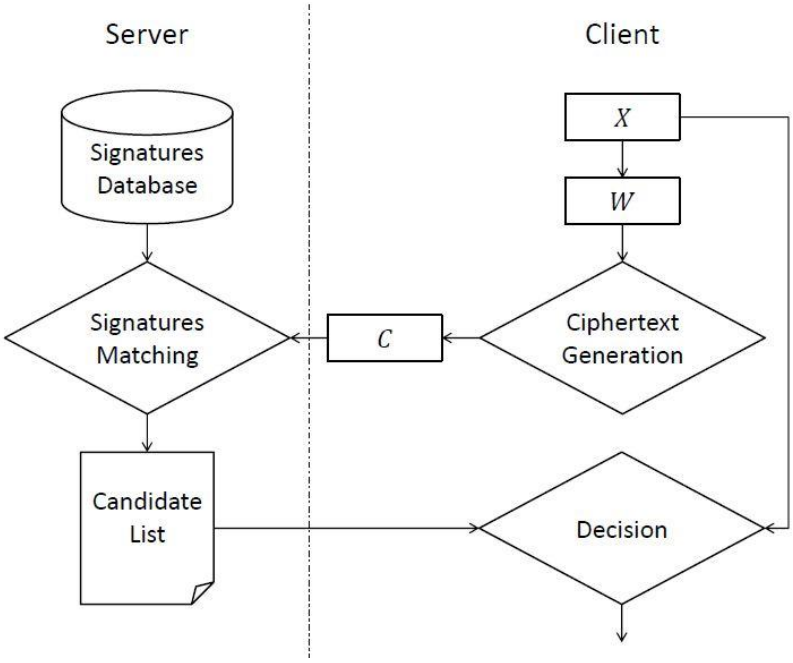
接著我們將說明比對的方法為何，在這我們會說明上述所提到的正確和失敗，何謂比對正確？何謂比對失敗？其關鍵字比對方法如下圖所示。首先先比對 signaturei 和 datai 關鍵字為 content 的部分，將這兩部分後面那一串二進位的值做 xor 的運算，會得到一個 Ki=Ki,a||Ki,b||Ki,c 的值，假設 h(Ki,a)前面的 s bits 和 Ki,b 的值一樣，則判斷為正確，反之則判斷為錯誤。將判斷正確的 rulei 和 Ki,c 放到候選者名單 L 中。



圖表二十五：關鍵字比對方法

目前這個 Server 程式會將收到的 Client 的 request 上傳到 Hadoop 的 HDFS 上，然後再去執行在 Hadoop 上的 www 這隻程式，當 www 程式執行完畢將結果留在 Hadoop 的 HDFS 上時，Server 才去將結果給取回來，回傳給對應的 Client 端。

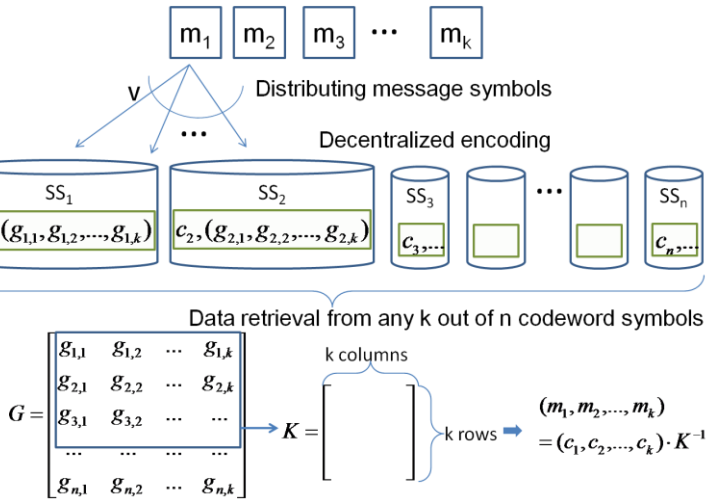
藉由前面雲端環境的建立和關鍵字資料庫的建立，此 client-server 程式根據 Song, Wanger, Perrig 的方法來完成保護隱私的功能，其正確執行流程如下圖所示。首先 Client 會收集封包 X，經過分析和轉換後將其長度固定為 W，利用 Chiphertext Generation 這步驟將 W 加密為 C 上傳到 Server；Server 處理收到的 C 後，經過 Signatures Matching 去比對 C 和 Signatures Database，若有符合的則放到 Candidate List 傳回給 Client；最後 Client 收到 Candidate List 後經過 Decision 的步驟盼端先前上傳的封包是否為一個惡意的攻擊。



圖表二十六：Client-Server 執行流程

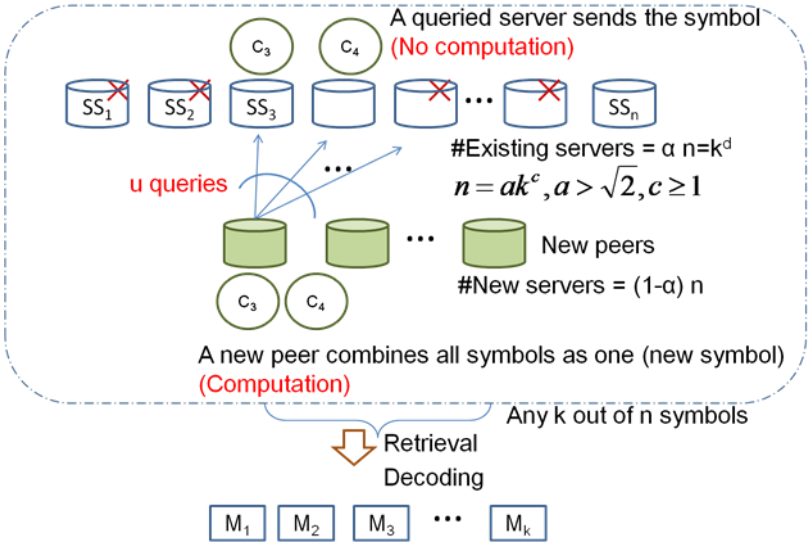
□ 非集中式雲端儲存系統中系統修復機制

我們提出的非集中式修復機制架構如下圖所示：



圖表二十七：非集中式修復機制架構

我們考慮當 n 個伺服器中的(1-α)n 個發生錯誤，系統新增了 α 個儲存伺服器進來。每個新的伺服器隨機地向 u 個舊的伺服器索取資料，舊的伺服器收到要求後會直接以儲存的資料回覆，新的伺服器為每份收到的資料隨機取一個係數並對所有收到的資料進行線性組合，最後僅將組合的值儲存下來。下圖為一個簡單的示意圖。



圖表二十八：非集中式修復機制

參數 u 值是系統中重要的參數，當 u 值設定的很高時，每個新的伺服器可以獲得較多的資料以取代發生錯誤的伺服器，但在連線數量上也就造成了較大的成本，另一方面來說，若 u 值設定得比較小，雖然降低了連線數量帶來的成本，但是新的伺服器有可能因為獲得得資料量太少而無法達到修復的功能。我們透過隨機程序的理論分析 u 值並且得到對 u 值的設定建議值，我們將結果分項表示於下圖中。

Theorem 1. *Let $n = ak^c$ for some constants a and c, where $c \geq 1$. Let the number αn of old servers be k^d, where $\alpha < 1$ and $d > 1$. Let the system be repaired by our repair mechanism with $u = k$. Consider the event of a successful data retrieval that k randomly chosen servers from new and old servers recover a message. The probability of a successful retrieval is at least $1 - \frac{2k}{p} - o(1)$.*

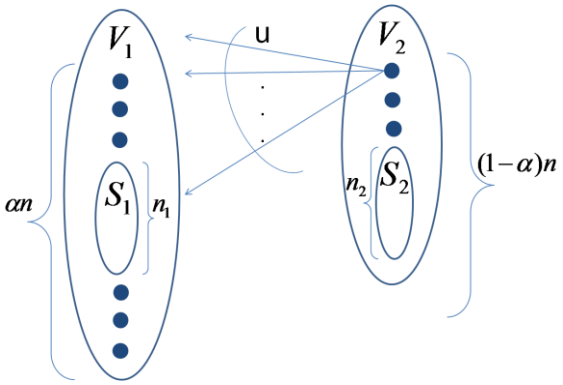
Theorem 2. *Let $n = ak^c$ and $\alpha n = k^d$ for some constants a, c, α, and d, where $c \geq 1$, $\alpha < 1$, and $d > 1$. Let the parameter u be set such that*

$$u \geq \min\{k, \max\{\frac{2k}{(d-1)\ln k}, \left(\frac{k}{(d-1)\ln k} + \frac{d}{d-1}\right)\}\}$$

After the system is repaired by our repair mechanism, the probability of a successful retrieval is at least $1 - \frac{2k}{p} - o(1)$.

圖表二十九：定理

證明這項結果的過程是本研究的重要貢獻之一。證明的方法則是先將修復機制中的資料流轉換成一個隨機 bipartite 圖來討論，圖中的點集合 V_1,V_2 為新的伺服器群(V_2)與舊的伺服器群(V_1)，當一個新的伺服器詢問一個舊的伺服器時，就形成了一個邊，所以此圖具有隨機的邊，如下圖所示。



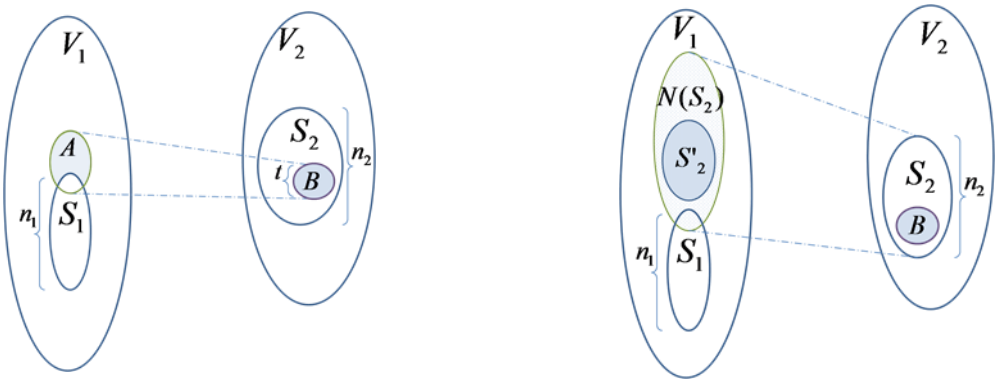
圖表三十：證明的方法

我們要探討的是資料能夠正確取回的機率，於是我們將一個使用者詢問的 k 個伺服器轉換為 k 個圖中的點，分別落在兩個點集合中的子集合 S_1 與 S_2，且將子集合的大小表是為 n_1 與 n_2，若此 k 點都落於 V_1，則依賴原本系統的容錯能力能夠保證資料可以被取回，但是當部份的點落到 V_2 時，要怎麼找到好的條件來使資料可以被取回呢? 令 $N(S_2)$表示 S_2 點集合在 V_1 中的鄰居點集合，我們觀察到一個充分與必要的條件來保證資料可以被取回:$N(S_2)\backslash S_1$ 與 S_2 有 maximal matching。觀察到這個條件之後，我們就可以使用著名的 Hall's Theorem 來分析此隨機圖中滿足該條件的機率。我們使用本文中的符號重新詮釋 Hall's Theorem 如下圖所示。

Lemma 2. (Hall's Theorem) *If and only of for any subset $B \subseteq S_2$, the number of neighbors of B in $V_1 \setminus S_1$ is no less than the size of B, i.e., $|N(B) \setminus S_1| \geq |B|$, where $N(B) \subseteq V_1$ is the set of neighbors of B, there exists a maximal matching from S_2 to $V_1 \setminus S_1$.*

圖表三十一：Hall's Theorem

按照重新詮釋的 Hall's Theorem，我們要估計的是不存在任何一個點集合 B 的機率，於是我們使用 union bound 高估存在任何一個點集合 B 的機率，再計算出不存在任何一個點集合 B 的機率的低估值，我們將 B 的角色表示於下圖中。對於存在任何一個點集合 B 的機率，我們則再次運用到對 B 的鄰居(A)個數進行 union bound 的技巧。我們將 A 的角色表示於下圖中。



圖表三十二：角色表示圖

透過這個條件，我們將可以取回資料的最小機率值設定在 1-k/p-o(1)，反推回對 u 值的設定條件如下不等式所示。

$$u \geq \min\{k, \max\{\frac{2k}{(d-1)\ln k}, \left(\frac{k}{(d-1)\ln k} + \frac{d}{d-1}\right)\}\}$$

為了了解這個設定值在實際系統中的影響，我們引用了實際大型分散式系統中紀錄並統計出來的數值，進行數值的分析。我們引用了系統的伺服器數量 n 以及平均每天會發生錯誤的數量 f，引用的數據呈列在下表中。

Trace	PlanetLab	Microsoft PCs	Skype	Gnutella
n: average number of nodes	303	41970	710	1846
f: fraction of failed node per day	0.017	0.038	0.12	0.3

我們首先考慮不同的 k 值，設定了不同的 d 值(正常運作的伺服器數量為 $\alpha n=k^d$)，計算出能符合不等式的最小整數 u 值，詳列於下表中。

$k=4$					
d	2	3	4	5	6
u	3	3	3	3	2
k^d	16	64	256	1024	4096

$k=8$				
d	2	3	4	5
u	6	4	3	3
k^d	64	512	4096	32768

$k=16$				
d	2	3	4	5
u	8	5	4	3
k^d	256	4096	65536	1048576

我們將計算出的數據佐以實際系統中的錯誤狀況估計出每個不同系統中能夠不進行修復作業而仍能維持服務的天數整理如下表。

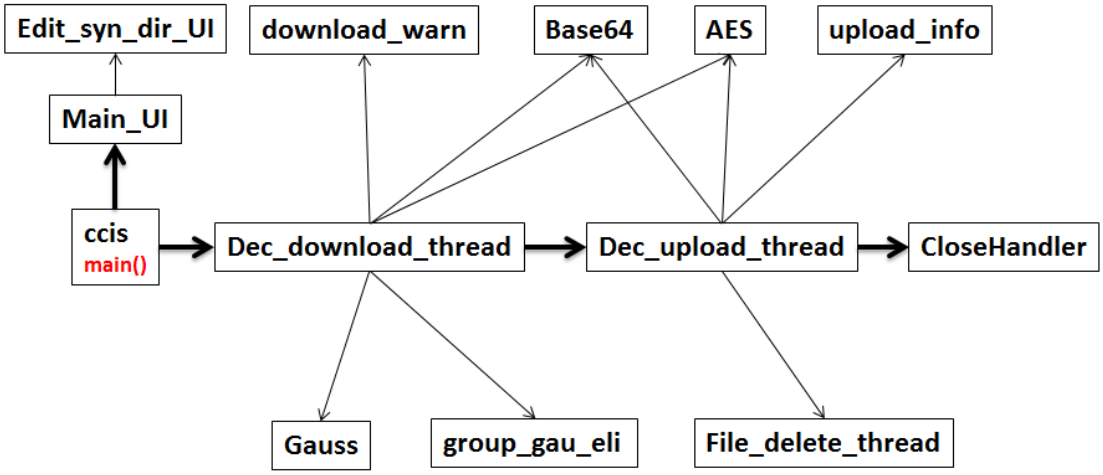
Trace	PlanetLab		Microsoft		Skype		Gnutella	
n	303		41970		710		1846	
f	0.017		0.038		0.12		0.3	
k	4	8	4	8	4	8	4	8
u	3	6	3	3	3	4	3	4
αn	16	64	16	4096	16	512	16	512
Survival duration (days)	47	39	26	23	8	2	3	3

由表中可以看到，如果在一個系統中具有 41970 個儲存伺服器，且平均每天會發生的錯誤量為 0.038，當參數值 k 設定為 4 時，系統可以在不進行修復的狀態下維持功能 26 天，若進行修復作業，則每個新增的儲存伺服器要向 3 個運作中的伺服器索取資料。

□ 使用者端資料加密容錯編碼之處理系統

使用者資料在上傳至雲端儲存之前，需要利用加密技術保護資料的內容；儲存的資料為了達到足夠的可靠性，雲端伺服器大都採用備份的方式來分散風險，分散式容錯編碼能夠提供更好的可靠性以及更經濟的儲存方式。現在第三方的雲端儲存空間相

當熱門，例如：Dropbox 和 SkyDrive，使用者可以上傳他的資料，並且在其他電腦上存取同步資料。雲端儲存服務通常只提供儲存空間，不具有資料運算能力，使用者只能自行保護資料的隱私性與可靠性，我們的系統利用 AES 加密技術保護資料隱私性，並且利用分散式的容錯編碼保障資料的可靠性。



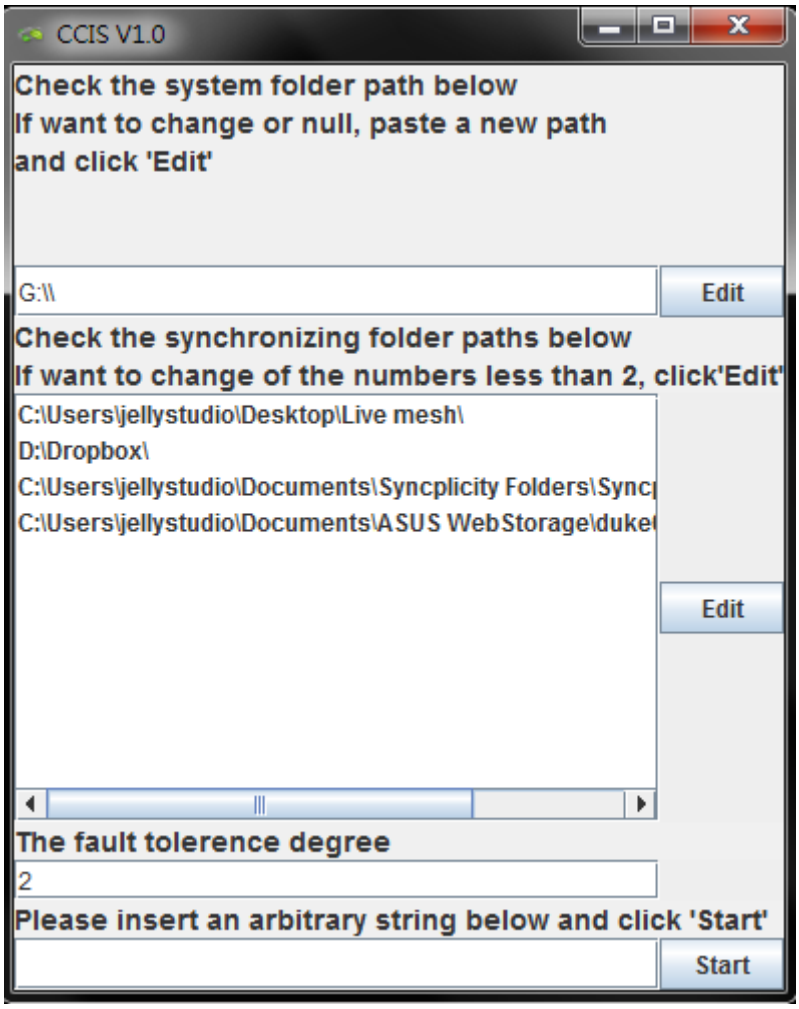
如上圖，我們系統在各類別執行的流程，主要的 main function 在 ccis.java 裡，由 ccis 去新增 Main_UI 的 thread 來產生使用者介面，讓使用者在系統一開始時可以編輯包括像系統資料夾路徑，共享資料夾路徑，還有一個隨機的字串。如果使用者需要的話，會產生 Edit_syn_dir_UI 的 thread 來編輯共享資料夾的路徑。

當使用者正確輸入了三個參數之後，我們系統會進入下個階段，呼叫 Dec_download_thread.java 類別的裡面的 function 來進行檔案的解密與解碼。當進行檔案的解密與解碼時，會需要利用五個不同的類別，Base64, Gauss, AES, group_gau_eli 所提供的 function 來進行檔案還原的計算。如果解密過程中發生了例外或是錯誤，這會產生一個 UI 的 thread 來通知使用者。當全部的檔案運算完畢之後，無論成功或失敗，會進入下一個階段。

接著系統會產生一個 Dec_upload_thread 的 thread 來執行無限迴圈，用來偵測是否以系統資料夾內的檔案需要上傳或刪除。在這個物件中，需要用到七個類別，分別是 Base64, Gauss, AES, group_gau_eli, File_delete_thread, upload_info, CloseHandler。在這個階段我們系統就是依需求來呼叫不同類別的 function，但在這階段開始時，我們系統也會產生一個 UI 的 thread upload info，告知使用者目前系統的狀況。並且處理 upload_info 裡的 exception listener 讓使用者關閉此 UI 後，我們系統將會關閉。

[class ccis]

首先，先從 main()內部來看，系統一開始會進行一些前置作業，先將一些上次執行的紀錄檔案刪除，目的只是為了降低後續系統執行時產生 bug 的機率。接著我們系統會產生 main_UI 的物件，名稱為 a，此時系統的 main thread 會用一個無限迴圈來等待使用者在執行 UI 後寫入一個隨意的字串到"Authenticate.txt"。



如上圖，當使用者輸入完四種資訊之後，並按下右下角的 Start botton，物件 a 會執行將系統資料夾路徑寫入"repository_dir.txt"，將共享資料夾路徑寫入"sync_dir.txt"，並將任意字串寫入"Authenticate.txt"。系統的 main thread 判斷了"Authenticate.txt"不為空之後，就會繼續往下執行。

在本系統中，不同物件的傳遞資訊是利用 file 的 input 與 output，例如某 a 物件寫入系統資料夾路徑，某 b 物件等到需要系統資料夾路徑時，再去讀取"repository_dir.txt"。或許有更好的方式來減少 file I/O 所造成的時間浪費。

在完成了四個資訊的輸入之後，我們系統會繼續進行前置作業。包括在系統資料夾路徑上產生一個系統資料夾"ccis_repository"以及在使用者指定的各個共享資料夾產生資料夾"@ccis_repositroy"。

接著會根據使用者指定的系統共享資料夾來進行遞迴的檢測，function 名稱為 search_data_dec()。在這個 function 裡面最主要的動作是針對每個共享資料夾內部的檔案與資料夾進行分類與整理。因為每個共享資料夾內都可能有檔案與子資料夾，所以先產生兩個 String array，file_record 與 dir_record。用記錄目前這個子資料夾內，有多少檔案與資料夾。

這邊作法是只要出現一個就會納入列表，意思就只要一個共享料夾有檔案或子資料夾，但我的系統依然會納入列表內，所以即使 codeword symbol 數量是低於 threshold，我程式之後也會將該檔案做解密與解碼，再讓解密與解碼來判斷是否可以

成功的還原資料。所以如果要更精簡運算時間的話，這部分是可以省略的，檔案 codewode symbol 數量低於 threashold 的將直接省略。

當系統整理出系統資料夾子子資料夾內的 file_record 與 dir_record，我們先依據 file_record 的內容依序處理，將處理的檔名，系統資料夾內相對路徑，還有共享資料夾內的相對路徑丟給 Dec_download_thread 類別內的 decrypt_a_file() function 運算。

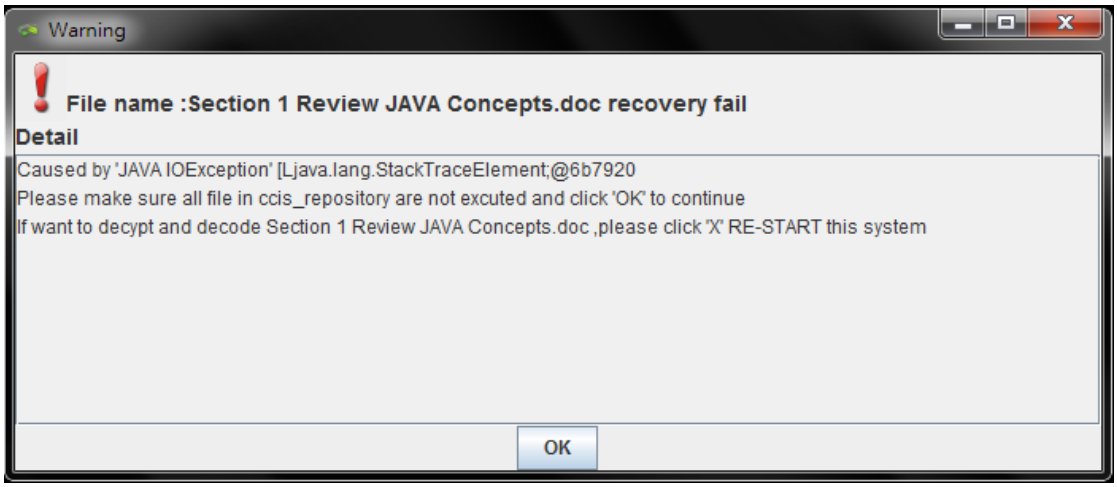
[class Dec_download_thread]

當執行到這個類別時，我們系統已經確定要處理哪個檔案，所以在這個類別主要是處理如何將檔案的 codeword 轉換成原始檔案。

系統再還原檔案的做法是，根據使用者所指定的共享資料夾內檔案來還原檔案，所以當初加密檔案時使用者指定了若干個共享資料夾，如果要還原時，使用者必須指定"相同的"共享資料夾，包括數量。因為在一開始 main_UI 指定的共享資料夾個數會影響到後續的運算參數，所以即使可以允許 k 個 codeword symbol 遺失，使用者還是必須完整的指定出當初加密編碼檔案時所指定的共享資料夾，但是順序是可以不一樣的。基於這個特性我認為可以做得更佳的有彈性，看是否有需要可以讓使用者少指定一個共享資料夾，我們系統可以判斷是其中一個 symbol 遺失，依然可以將檔案還原。

那麼在 Dec_download_thread 類別還原檔案的主要計算在 decrypt_a_file() function 內，依需求去呼叫 base64, AES, Gauss, group_gau_eli 內的 function()來計算，如果有計算錯誤發生的話，會新增一個 download_warn 物件 UI 來提示使用者，詳細的內容會在後面的段落闡述。

在還原檔案的計算過程，在系統資料夾內的檔案是不可被開啟的，這問題會出現在 office 的系列。這問題目前我沒有辦法解決，只能用錯誤訊息來通知使用者。



上圖是錯誤訊息的 UI，內容會依據不同的原因而顯示給使用者。目前仍未將可能的錯誤原因做完，所以 UI 訊息的補充也是可以加強的部分。

在 decrypt_a_file()內，執行的順序是先將當初加密檔案的 encryption key 還原，然後再利用 encryption key 來還原原本的資料。所以為了還原該檔案的 encryption key 的資訊，系統會先去使用者指定的共享資料夾內搜尋相關的 codeword symbols，也就是在該路徑的資料夾內，找尋 原始檔名+".$#_2"+"原始副檔名。

之後我們系統會利用隨機的方式，找尋 k 個適合的 encryption key symbol 來當作我們的解密與解碼 encryption key 所需要的 symbols。我們系統會以最多一千次的隨機搜尋，來找到哪 k 個 symbol 的內容可以計算生成矩陣的反矩陣。當一千次搜尋完沒有結果時，就會判定這個檔案還原失敗。不過這部分可以利用別的搜尋法來代替，或許可以減少 false negative 的機會。在隨機選取出 k 個 codeword symbols 之後，會利用 group_gau_eli 類別內的方法 det 來判斷我們選取的內容所產生的矩陣的 determinant 的值來判斷是否有反矩陣，如果有的話就把這個 k 個 codeword symbols 位置記錄起來，之後就選取這 k 筆資料。如果 determinant 為 0 的話，代表沒有反矩陣，所以只能在繼續搜尋下一組隨機的 k 個 symbols。

當確定好 k 個 encryption symbols 的位置之後，就去將 k 的 symbol 的資訊讀取出來，並且先計算其反矩陣：

Element[][] inv2 = group_gau_eli.*inv*(_k_mat_ele) ;

在 group_gau_eli 裡面的 inv 以及 Gauss 的 gau functions 都是利用高斯消去法來計算出反矩陣，這邊基本上是沒有什麼問題的，另外在 Gauss 裡還有提供分數的加法，乘法，通分，約分，還有矩陣相乘等 function 運算。

接著是依造 protocol 來實作解密與解碼的函示

```
/**
 * 利用 codeword 的第二個資料產生 hid
 */

Element hid = pairing.getG1().newElement() ;
byte[] hid_temp    = new byte[152] ;
for( int j=0 ; j<152 ; j++ )
{
      hid_temp[j] = codeword[0][152+j] ;
}
hid.setFromBytes(hid_temp) ;



/**
 * 計算 hidx
 */
Element sk = read_sk() ;
Element hidx = pairing.getG1().newElement() ;
hidx.setFromBytes(hid_temp) ;
hidx.powZn(sk) ;

/**
 * Decryption
```

```
 */
Element[] w = new Element[share_count] ,
            A = new Element[share_count] ,
            B = new Element[share_count] ,
            numerator = new Element[share_count] ,   //分子
            denominator = new Element[share_count] ;   //分母
//初始
for( int k=0 ; k<share_count ; k++ )
{
    w[k] = pairing.getGT().newElement() ;
    A[k] = pairing.getG1().newElement() ;
    B[k] = pairing.getGT().newElement() ;
    numerator[k] = pairing.getGT().newElement() ;
    denominator[k] = pairing.getGT().newElement() ;
}

for( int k=0 ; k<share_count ; k++ )
{
    byte[] A_temp = new byte[152] ;
    byte[] B_temp = new byte[152] ;

    for( int l =0; l<152 ; l++ )
    {
        A_temp[l] = codeword[k][l] ;
        B_temp[l] = codeword[k][304+l] ;
    }
    A[k].setFromBytes(A_temp) ;

    numerator[k].setFromBytes(B_temp) ;
    denominator[k] =    pairing.pairing(A[k], hidx);
    w[k] = numerator[k].add(denominator[k].invert()) ;
}

/**
 * 現階段的 w[i] 是 session key 的 linear combination
 * 所以跟反矩陣做運算。
 */
Element[] result_temp = new Element[share_count] ;
for( int k=0 ; k<share_count ; k++ )
    result_temp[k] = pairing.getGT().newZeroElement() ;
```

```
for( int j=0 ; j<share_count ; j++ )
{
    Element[] point_temp = new Element[share_count] ;

    for( int k=0 ; k<share_count ; k++ )
        point_temp[k] = w[k].duplicate() ;


    for( int k=0 ; k<share_count ; k++ )
    {
        point_temp[k].powZn(inv2[k][j]) ;
        result_temp[j].add(point_temp[k]) ;
    }
    //System.out.println("result_temp["+j+"]="+result_temp[j]) ;
}

String EncryptionKey = new String("") ;
for( int k=0 ; k<_k ; k++ )
    EncryptionKey += result_temp[k] ;
```

上述的步驟就是將 encryption key 還原。字串變數名稱是 session_key。

當還原了 encryption key 後，接下來我們系統要執行將每個檔案的 shares 還原然後再組合起來。所以我們還原檔案的順序是依 shares 的編號γ，其中γ是在加密與編碼前，會將原始的檔案利用 File 類別的 FileChannel 函式，將原始檔案以 10MB 的大小切割成若干個 shares。然後再每個 shares 依序進行加密與編碼的動作，最後在寫入 codeword 檔案到共享資料夾時，再依序從 0 到$\gamma - 1$編號，例如 123.$#_1_0.mp3 和 123.$#_1_1.mp3，編號是在副檔名之前的最後一個數字。所以我們系統再還原檔案的時候，也是根據編號，依序將每個 shares 還原，在將檔案透過 FileChannel 合併

在沒有完整檔案之前，我們系統無法判斷一個完整的檔案原始大小為多大，所以我們利用 codeword symbol 的γ編號來判定一個檔案在當初切割時，被切割了多少份。判斷的方式是，當我們系統決定要進行解密與解碼該檔案時，系統會去使用者指定的共享資料夾內找尋相關的 encryption key 和檔案的 codeword symbol。然後在根據檔案的 symbol 檔案中，編號最大的數字為多少，藉此來判定該檔案粗略的大小，接著在依每個編號去將檔案的 shares 逐步完成。

而我們系統提供了一個及彈性化的容錯能力。只要每組 shares 的 codeword symbols 的還有相同資料夾內的 encryption key 的 codeword symbol 都存在且總數量大於等於 k 個，我們系統就可以將該 shares 還原。因此，我們系統提供的容錯能力並不局限在當 erasure 發生在相同的 (n-k) 個網路儲存服務系統。

我們系統會依造 codeword 的編號去將整個檔案還原，從 0 到**Y-1**，將 codeword 還原成子檔案，再將每個子檔案組合起來。所以針對每個 codeword，先透過 BigInteger 函示將 codeword symbol 檔案從 byte array 轉成 BigInteger

```
for( int k=0 ; k<_k ;k++ )
codeword_temp[k] = new BigInteger (data_shares[k]);
```

接著進行矩陣相乘的動作來達成 decoding：

```
BigInteger[] word = Gauss.matrix_mul(codeword_temp, inv) ;
```

將 word 再轉回 byte array 並做 base64 decode

```
for( int k=0 ; k<_k ; k++ )
{
data_shares[k] = word[k].toByteArray() ;
data_shares[k] = Base64.decode(data_shares[k]) ;
}
```

再來將每個 data_shares 組合成一個較大的 byte array file，並利用 AES-128 來解密

```
file = AES.getdecrypt(EncryptionKey, file) ;
```

在這個步驟做完之後，我們已經完成了每個檔案的子檔案的還原動作了，接著我們在系統資料夾內產生一個資料夾叫做 tmp2 的資料夾，將之前計算出來的 file 寫入 tmp2 資料夾內存成 tmp 檔，最後將所有檔案利用 FileChannel 組合起來並寫到系統資料夾內相對應的位置。所以我們系統是利用這個方式來處理檔案太大的問題。

[class ccis]

當處理完一個檔案的時候，會看 file_record 內是否還有尚未處理的檔案，並依上述的方法完成之後，接著要處理 dir_record。dir_record 裡面紀錄的是共享資料夾內有的出現的子資料夾名稱。因此針對每個 dir_record 裡的紀錄，我們系統處理的方式是依造 dir_record 紀錄的名稱在系統資料夾內相對路徑也產生相同名稱的資料夾，並遞迴執行 search_data_dec()進到子資料夾內，利用同樣的方式將共享資料夾內的檔案與資料夾還原到系統資料夾內。

最後是執行，sync_log()，利用系統資料夾內還原過後的檔案進行紀錄檔的編譯，利用遞迴的方式將系統資料夾內的檔案與資料夾資訊都編寫到對應的紀錄檔。

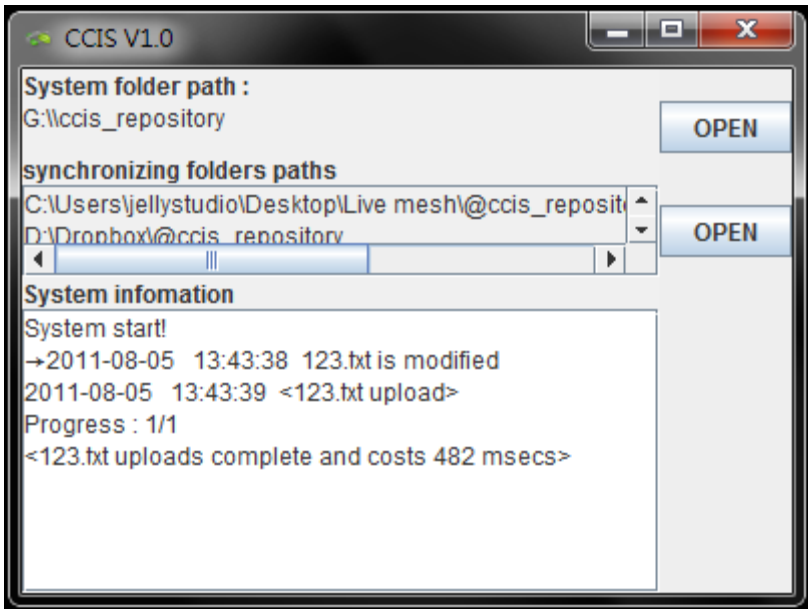
當系統完成了所有檔案的還原並編寫好紀錄檔之後，此時後產生一個 thread 來執行偵測的動作，我們系統便進入下一個階段。。

```
Thread det_up = (Thread) new Dec_upload_thread() ;
det_up.start() ;
```

[class Dec_upload_thread]

當 main thread 產生了新的 thread det_up，在這新的 thread 會執行 run() function 的內容。在 run()裡面，首先會先物件 upload_info 的 UI，名稱是 a。產生這個 UI 的

目的是希望讓使用者可以在後續的使用過程中，可以看到系統執行的資訊，包括檔案是否有處理成功等等。



這方面的技術是在系統計算的過程，將各種資訊的字串寫到 UI 的元件裡，並顯示出來：

```
pre_msg = a.getJTextArea5().getText() ;
pre_msg = pre_msg +"System start!" ;
a.getJTextArea5().setText(pre_msg) ;
pre_msg = "" ;
a.sbar.setValue(a.sbar.getMaximum()) ;
```

其中，getJTextArea5()即是 UI 中 System information 的元件部分，a.sbar.setValue(a.sbar.getMaximum())這句的意思是讓 UI 的滾輪可以隨時都隨著資訊的增加滾到最下方新資訊的地方。

之後讀取基本的資料，包括像系統資料夾的路徑，與共享資料夾的路徑，當系統完成這個步驟之後，會打開系統資料夾給使用者存取：

```
File open_dir = new File(Up_dir) ;
if ((open_dir.exists())&&(open_dir.isDirectory())){
try {
  Process process = Runtime.getRuntime().exec("explorer "+open_dir);
  System.out.println(process) ;
} catch (IOException ioe) {
                    //do something
}
}
```

接著 det_up 這個 thread 會執行無限迴圈，用來做之後的偵測。分別不斷執行以下的函示：

1. check_log()

   這個函式主要的目的是要偵測是否有新檔案新增到系統資料夾。

2. auto_recovery()

   這個函式主要的目的在於偵測系統執行時，共享資料夾內是否有 symbol 檔案遺失。

3. File_delete_thread.det_del()

   偵測系統資料夾內原始的檔案是否有遺失。

4. det_edit()

   偵測系統資料夾內原始檔案是否有被修改過。

5. check_at_ccis_folder()

   為了避免共享資料夾內"@ccis_repository"的資料夾發生遺失導致後續處理的路徑不一致，所以利用這個函式來檢查共享資料夾內"@ccis_repository"

6. clean_tmp2()

   這個函式主要是在處理在解密與解碼的過程，我們系統會產生 tmp 檔案，而計算結束之後，理應會將 tmp 檔給刪除。但有時候不知原因刪除會失敗，所以透過這個函示來不斷的檢測，如果還有暫存的 tmp 檔，就一直執行刪除的動作。直到將 tmp 檔都刪除完畢為止。

而在處理檔案的加密與編碼的部分，是在函式 encrypt_a_file()內執行，因為運算的需要，還需要利用函示 data_share_compute()，gama_compute()，alpha_compute()，generator_matrix()，generate_session_key()等的函式，這在後面的部分會詳細的說明。

我們接下來的報告會詳細介紹關於 encrypt_a_file()函式的內容，接著再補充說明各種偵測是如何實做出來的。

encrypt_a_file()函式需要若干的參數，包括即將要進行加密編碼的檔案的 File class 物件，還有系統資料夾內的相對路徑，以及還有共享資料夾的路徑。在一開始完成了前置的動作之後，接下來要運算檔案的大小：

```
String src_file = new String(upload_file.getAbsolutePath()) ;
int split_num=0 , remain_len=0;   // 計算總共要切割的長度與最後一塊的長度

try{
FileInputStream src = new FileInputStream(src_file) ;
      split_num = src.available()/10485760+1 ;
      remain_len = src.available()%10485760;
      src.close() ;
}catch(IOException s)
{
return false;
} catch (Exception e) {
return false;
}
```

我們這個步驟是計算每個檔案以 10MB 大小為切割時，會切割出幾塊。其中變數 split_num 就是紀錄總共會切割出幾塊，remain_len 是餘數的部分。

接著是在各個相對應的 groups 上宣告資訊：

```
int index_temp = Math.abs(upload_file.getName().hashCode())%100 ;
Element[] hidr = new Element[_k] ;
for(int k= 0 ; k<_k ;    k++)
{
hidr[k] = pairing.getG2().newElement() ;
}

Element hid = pairing.getG1().newElement()
, g_temp = pairing.getG1().newElement() ;
Element[] pub_key = new Element[_k]
                    , paired_point = new Element[_k]
                    , garmma = new Element[_k]
                     , gr = new Element[_k] ;

for( int k=0 ; k<_k ; k++)
{
gr[k]= pairing.getG1().newElement() ;
pub_key[k] = pairing.getG1().newElement() ;
paired_point[k] = pairing.getG2().newElement() ;
garmma[k] = pairing.getGT().newElement() ;
garmma[k].setToZero() ;
}
```

其中_度為 k 是 word 的。在上述的步驟就是在各個群上宣告變數，其中要說明的是，在 jPBC 中的函式，是將 G1 上的點和 G2 上的點對應到 GT 上，$\tilde{e}(G1, G2) \rightarrow GT$，由於我們載入的是 symmetric pairing 的資訊，所以其實在實作中，G1 會等於 G2。而我們系統宣告的原則在於，在原 protocol 的 pairing 函式裡，左邊的元素就宣告成 G1，右邊的元素就宣告成 G2，pairing 過去到 GT 上。

接著處理運算完得 codeword symbol 檔案的檔名：

```
update_name    = new String("") ;
update_key    = new String("") ;
for( int j=0 ; j<_k ; j ++ )
{
update_name += name_temp[j] ;
update_name += "." ;
update_key += key_temp2[j] ;
```

```
update_key += "." ;
}
update_name =    update_name + "$#_1." + ame_temp[name_temp.length-1];
update_key =    update_key + "$#_2." + key_temp2[key_temp2.length-1];
```

這邊設計得邏輯是，檔案的 codeword 就是編號 1，encryption key 的 codeword 就是編號 2。

接下來要處理檔案的部分，我們接下來要處理的部分是要利用 JAVA File class 的 file channel 的方法來達到切割大檔案的目的。我們將一個大的檔案以約 10MB 的大小來切割檔案，將每個檔案切割成若干個 tmp 檔的小檔案，每個 tmp 檔都會有編號，從 0 到 split_num-1：

```
for( int j=0 ; j<split_num ; j++ )
{
tmp_file_dir[j] = new String(tmp_dir) ;
tmp_file_dir[j] = tmp_file_dir[j] +String.valueOf(j) +".tmp" ;
}
try{
FileInputStream src = new FileInputStream(src_file) ;
FileOutputStream[] write = new FileOutputStream[split_num] ;
for( int j=0 ; j<split_num ; j++ )
      write[j] =    new FileOutputStream(tmp_file_dir[j]) ;

FileChannel read_src = src.getChannel() ;
FileChannel[] write_tmp = new FileChannel[split_num] ;
for( int j=0 ; j<split_num ; j++ )
      write_tmp[j] = write[j].getChannel() ;

MappedByteBuffer[] mbuf = new MappedByteBuffer[split_num];
for( int j=0 ; j<split_num ; j++ )
{
      if(j<split_num-1)
mbuf[j] = read_src.map(FileChannel.MapMode.READ_ONLY ,j*10485760 ,
10485760) ;
      else
mbuf[j] = read_src.map(FileChannel.MapMode.READ_ONLY ,j*10486760 ,
remain_len) ;
                  write_tmp[j].write(mbuf[j]) ;
}

src.close() ;
read_src.close() ;
```

```
for( int j=0 ; j<split_num ; j++ )
{
    write[j].close() ;
    write_tmp[j].close() ;
}
```

所以我們系統將任何格式，任何大小的檔案，都是用這個方法將其切割成 tmp 檔並暫存到"tmp"資料夾，之後處理檔案的加密與編碼都是處理每個檔案的 tmp 檔。

接著，前置動作完成後，要開始執行檔案的加密與編碼。我們系統針對同一個檔案都使用相同的 encryption key 來做加密的動作，因此即是檔案會切割很多份的 tmp 檔，還是使用相同的 encryption key 來當作 AES-128 加密的 key：

```
Element[] EncKey_temp = generate_session_key()   ;
String Encryption_key = new String("") ;
for( int k=0 ; k<_k ; k++ )
    Encryption_key += EncKey_temp[k].toString() ;
```

我們透過 generate_session_key 在 GT 上產生 k 個資訊。然後我們用 String 變數 Encryption_key 來記錄這些資訊的組合來當作 encryption key。

接著依造 protocol 來加密 encryption key 裡面每個資訊，以及產生一個 k×n 的生成矩陣並做編碼：

```
Element[] alpha =   alpha_compute(gr,generator_matrix) ;
        Element[] gama = gama_compute(garmma,generator_matrix) ;
```

最後要將結過寫入檔案內，因為 G1，G2 和 GT 的資訊轉換成 Byte array 的話長度為 152，而 Zr 上的資訊轉換成 Byte array 的長度為 23，所以要將 codeword symbol 寫入檔案，需要宣告一個 Byte array：

```
byte[][] codeword = new byte[_n][456 + 23*_k] ;
```

然後我們系統將所有的 encyption key 的 codeword symbol 依序寫到 Byte array 內：

```
for( int k=0   ; k<_n ; k ++ )
{
for(int l=0 ; l<152 ; l++)
{
codeword[k][l] = alpha[k].toBytes()[l] ;
codeword[k][152+l] = hid.toBytes()[l] ;
codeword[k][304+l] = gama[k].toBytes()[l] ;
}
for( int l=0 ;l<_k ; l ++ )
{
for( int m=0 ; m<23 ; m++ )
{
codeword[k][456+l*23+m] = generator_matrix[l][k].toBytes()[m] ;
```

}

}}

當完成了 encryption 初始的部分之後，接著就是要處理檔案的加密與編碼，我們依序將每個 tmp 檔讀取出來，針對每個 tmp 檔，透過 AES-128 的加密：

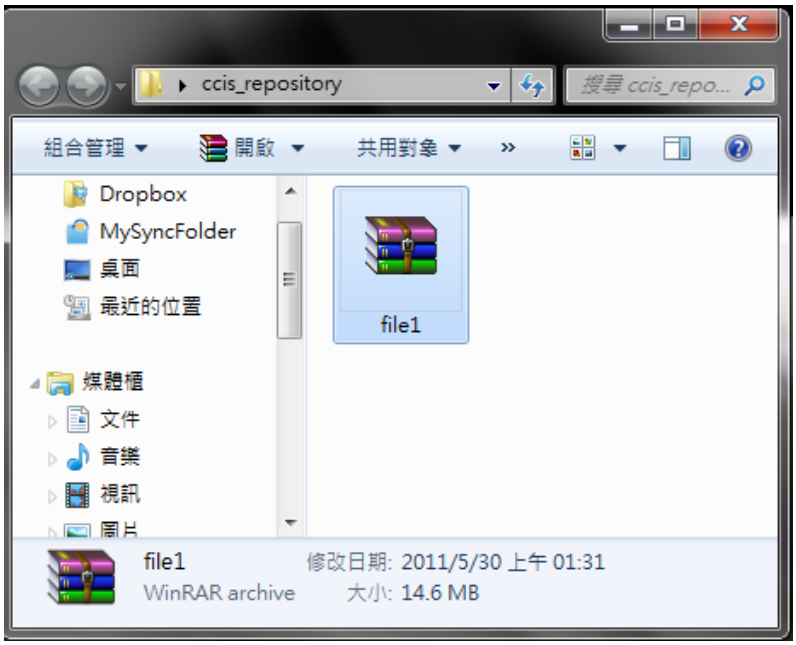
byte[] ciphtext = AES.*getencrypt*(Encryption_key, data) ;

關於 AES 的使用我們系統還有使用 CBC 以及 PKCS5 的 padding，所以檔案加密過後的檔案的大小膨脹會比較大，如果再加上後續的 Base64 編碼，則檔案會膨脹約原本的 4/3 倍，關於這點可以研究是否還有需要使用。

接著將檔案透過 Base64 encode 過後，再轉換成 BigInteger 來跟 BigInteger 版本的生成矩陣進行矩陣相乘的動作，來達到編碼的運算。最後將 n 個 symbol 轉換成 Byte array。

最後的動作，是要將檔案的 codeword 與 encryption key 的 codewords symbols 依序寫到使用者指定的共享資料夾內。使用者在 UI 所輸入的共享資料夾從上至下依序是 0 到 n-1，而所計算出來的 n 個 sumbols 也依造共享資料夾的順序寫入，E.G.第 0 個 symbol 寫到使用者指定的第一個共享資料夾內。而為了之後可以順利將正確的檔案取回，寫入檔案會有一定的命名規則，其中每個檔案只有一個 encryption codeword，命名規則是 *file name*.$#_2. *Extension* ，我們保留了原始檔案的檔名與副檔名，只在中間加上了".$#_2"，由於每個 symbol 要寫到不同的共享資料夾內，所以每個 symbol 可以檔名都一樣。而檔案的 codeword symbol 命名規則是 *file name*.$#_1_ i. *Extension*，其中 i 是編號，從 0 到 split_num-1。

結果如下圖：



我們將將一個 15MB 大小的檔案拖曳到系統資料夾內，我們系統針對 file1.rar 檔運算後，會將結果寫到使用者指定的共享資料夾內

因為我們命名有保留副檔名，所以在共享資料夾內，還可以看到 symbol 的檔案還有對應的作業系統產生的 icon。而因為檔案有 15MB 的大小，所以我們一開始在做檔案切割時，會將原始檔案切成兩塊，分別對這兩塊進行加密與編碼的運算並產生兩個不同的 codeword。

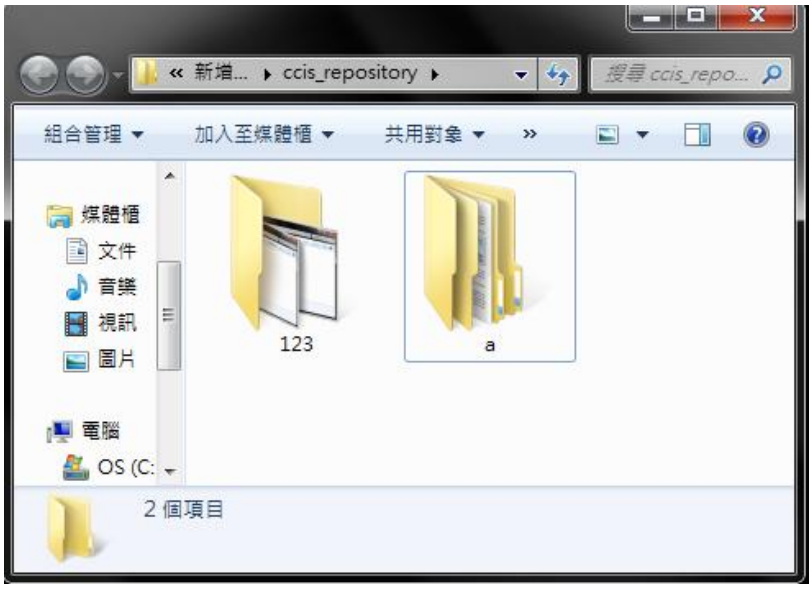
在介紹完加密與編碼檔案的方法之後，接下來就要介紹我們系統平常在執行的過程，是什麼樣的事件會觸發加密與編碼檔案。要觸發加密與編碼的時機在於

1. 新增檔案到系統資料夾
2. 系統資料夾內檔案有修改過
3. 共享資料夾內 symbol 檔案遺失

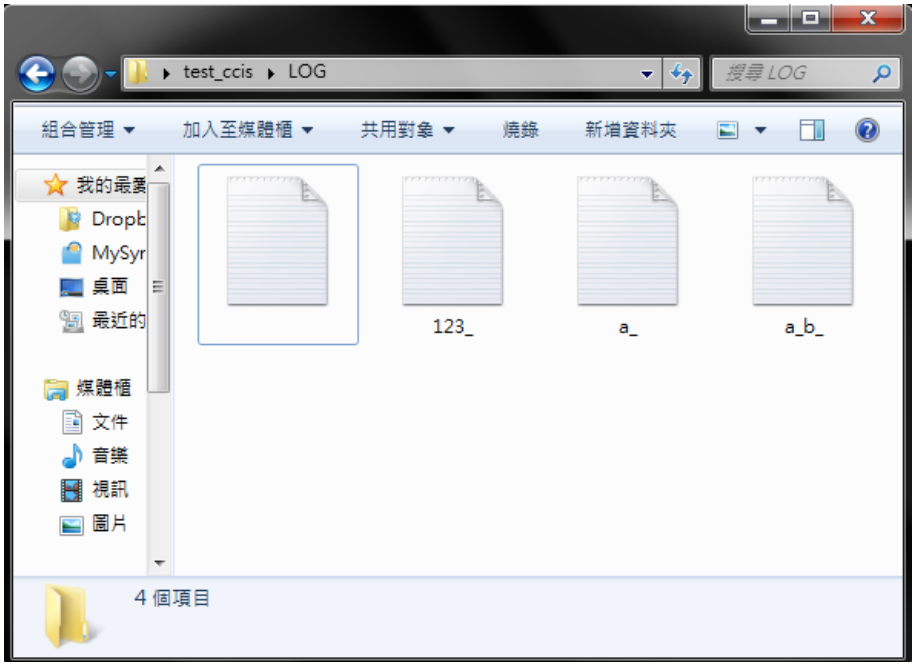
針對上述三個事件，我們系統利用紀錄檔的方法來比對系統資料夾內的檔案是否有更動過，所以接下來要介紹是如何實作出偵測的方法。

偵測新增檔案的函式主要是寫在 check_log()。這個函式主要是在比對記錄檔與系統資料夾裡面的檔案是否有出現差異。我們系統的記錄檔是儲存在與主 jar 檔程式的相同的路徑裡的一個資料夾，名稱為"LOG"。"LOG"資料夾內可能存放多個紀錄檔，每個紀錄檔紀錄的是系統資料夾內裡面其中一個子資料夾內檔案的資訊，包括檔名與上次修改時間。例如，系統資料夾內裡面有個檔案 A.txt，有個資料夾"B"，"B"裡存放一個檔案 C.txt。那麼"LOG"資料夾內就會又兩個紀錄檔，一個是無名的紀錄檔" .txt"

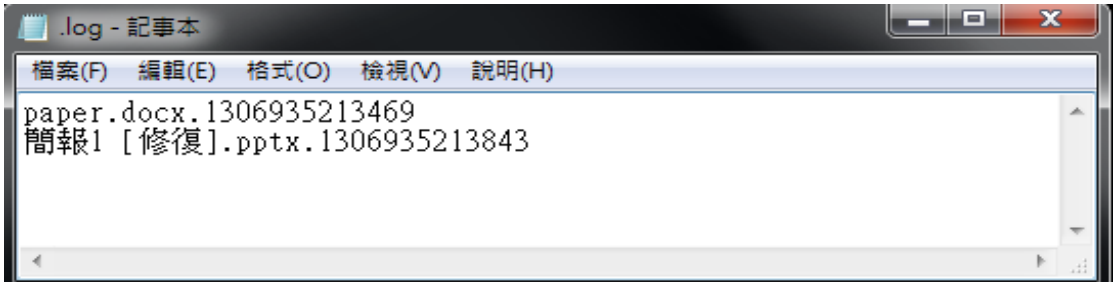
另一個是"B_.txt"。我們紀錄檔的命名規則就是利用系統資料夾內相對路徑來作為命名。



上圖是系統資料夾內的檔案跟資料夾，其中有三個資料夾，分別是"123"，"a"，a裡面還有一個資料夾"b"。



如上圖所以會產生四個紀錄檔。無名紀錄檔就是紀錄系統資料夾最上層的資訊。而"123.txt"，"a.txt"， "b.txt"就是紀錄相關的子資料夾的資訊。



如上圖，紀錄檔的資訊就是紀錄該子資料夾內的所有檔案，以及上次儲存的時間。

所以在 check_log()最主要的就是不斷的遞迴掃描系統資料夾內的子資料夾，並且讀取其相關的紀錄檔的資訊來做各種的比對。check_log()一開始就是讀取相關的紀錄檔並執行 parse。

如果使用者將隱藏檔新增至系統系統資料夾，我們系統也可以偵測得出來並做出加密與編碼的動作，那麼下次執行時，就會還原出一個非隱藏的檔案。針對保護使用者的意願，我們系統不執行隱藏檔。

因為我們系統支援使用者在我們系統執行的過程中可以編輯檔案，當修改過的檔案儲存之後，我們系統會偵測出且做出對應的加密與編碼。但是關於 office 的軟體，包括項 office 與 powerpoint 等等的，儲存時會額外產生一個檔案，例如檔名是"~"開頭",tmp"結尾。因為會產生一個額外的檔案，所以會造成我們系統以為有新的檔案新增到系統資料夾內，因此，我們系統會加密與編碼該新增的檔案。為了避免種情況，我目前的作法是將已知會產生的可能檔案先記錄好，如果有類似的格式的檔案新增，則不去處理他。那些新增的檔案過了一段時間後自己會消失，對整個系統資料夾與紀錄檔和共享資料夾並不會有造成不一致的現象。以下的程式碼就是在排除系統去運算隱藏檔跟 office 軟體產生的暫存檔。

```
int hidden_count = 0 ;
for( int i=0 ; i<dir_content.length ; i++ )
{
if( dir_content[i].isHidden())
{
      hidden_count ++ ;
}
}

int office_count = 0 ;
for( int i=0 ; i<dir_content.length ; i++ )
{
if( dir_content[i].getName().startsWith("~") &&
dir_content[i].getName().endsWith(".tmp") )
                  office_count ++ ;
}
```

```
for( int i=0 ; i<content.length ; i++ )
{
    if(dir_content[i].getName().equals("tmp2") || dir_content[i].isHidden() ||
(dir_content[i].getName().startsWith("~") && dir_content[i].getName().endsWith(".tmp"))
                || (dir_content[i].getName().endsWith("00"))
                || (dir_content[i].getName().endsWith("tmp"))
                || (dir_content[i].getName().startsWith("ppt") &&
dir_content[i].getName().endsWith(".tmp")))
        continue ;
    content[i] = dir_content[i].getName() ;
}
```

當系統資料夾裡子資料夾內的檔案的個數比紀錄檔還多時，parse 出是哪個新增的檔案或資料夾，並執行 directory_recur()。directory_recur()主要是針對使用者新增了資料夾或是檔案到系統資料夾，如果是資料夾的話，那麼我們系統的執行邏輯是，在使用者指定的共享資料夾內，也產生相同名稱的資料夾，然後在遞迴執行 directory_recur()，找到其子資料夾內的檔案，並將該檔案進行加密與編碼，結果將會寫到各個共享資料夾內其跟原始檔案相同相對路徑的資料夾內。所以當 check_log() 偵測出新增的檔案與資料夾時，就會來執行 directory_recur()將資料夾內的所有檔案與資料夾做適當的處理。

在 directory_recur()內，要處理的是檔案，則直接執行 encrypt_a_file()，並且重新編輯紀錄檔。如果是資料夾的話，則先編輯紀錄檔，然在各個共享資料夾內的相對路徑上產生相同名稱的資料夾，最後在遞迴到該子資料夾內。

接著要進行的是下一個偵測項目，是檢查共享資料夾內存放的各個檔案的 codeword symbol 檔案數量是否是有減少。檢測的方式是讀取系統資料夾的子資料夾內的所有檔案，根據檔案的大小我們可以算出正確的 codeword symbol 的個數，所以再依序去各個使用者指定的共享資料夾內統計檔案的個數。一旦有少的話，則會重新加密與編碼該檔案一次。

下一個測試的項目是檢查系統資料夾內的檔案數量比紀錄檔的少，這種案例我們設定是使用者想要刪除在雲端上的檔案，而又不想逐一地去各個共享資料夾內刪除檔案，所以直接透過我們的系統就可以將所有雲端上的相關檔案刪除。我們是在 File_delete_thread 的類別中實作將檢測檔案刪除的動作。檢測的方式與判斷是否有檔案新增的方法類似，只是這邊檢查是系統資料夾的子資料夾內的檔案數量比對應的紀錄檔的內容數量少時，我們就會判定使用者刪除了系統資料夾內的檔案，所以系統會去使用者指定的共享資料夾刪除相關的 codeword symbol 檔案。

雖然對整體影響些微，但也可以將 File_delete_thread 類別內的函示寫到類別 Dec_upload_thread 內。會分開寫其實沒有特別的涵義，只是當初設計後來有經過多次修改，才會出現變數名稱迥異與多餘類別的情況。

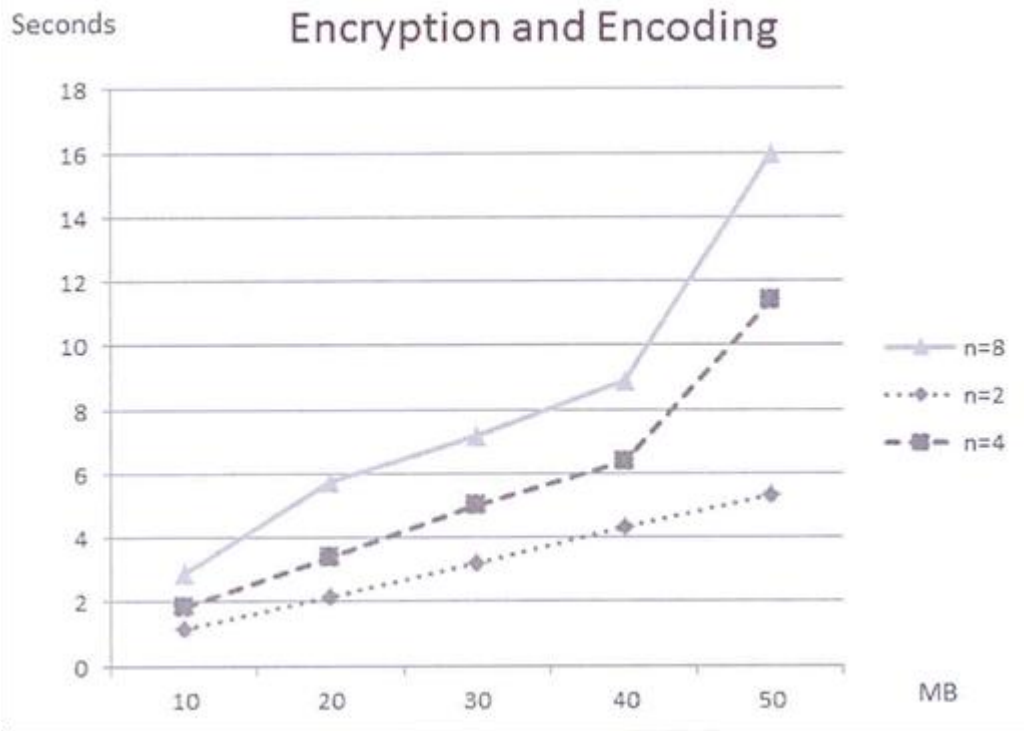
最後是偵測系統資料夾內的檔案是否有被修改過，判斷的方式依然是讀取系統資料夾內的各個子資料夾，然後再讀取其對應的紀錄檔，我們偵測系統資料夾內的檔案是否有修改過，是根據 JAVA File class 的函示 lastmodified()，此函式會 return 一個類

型為 long 的數字，我們系統在完成解密與解碼時，會先讀取每個檔案的 lastmodified()值並存到紀錄檔裡。所以做法與之前類似，一樣是遞迴的讀取每個系統資料夾內的子資料夾，然後利用 lastmodified()去讀取每個檔案的回傳值，並與紀錄檔的資訊做比較。如果系統資料夾的檔案有修改過並儲存後，其 lastmodified()的值會與紀錄檔的值不一致，所以就執行 encrypt_a_file()，結束後再更新新的 lastmodified()值到紀錄檔。

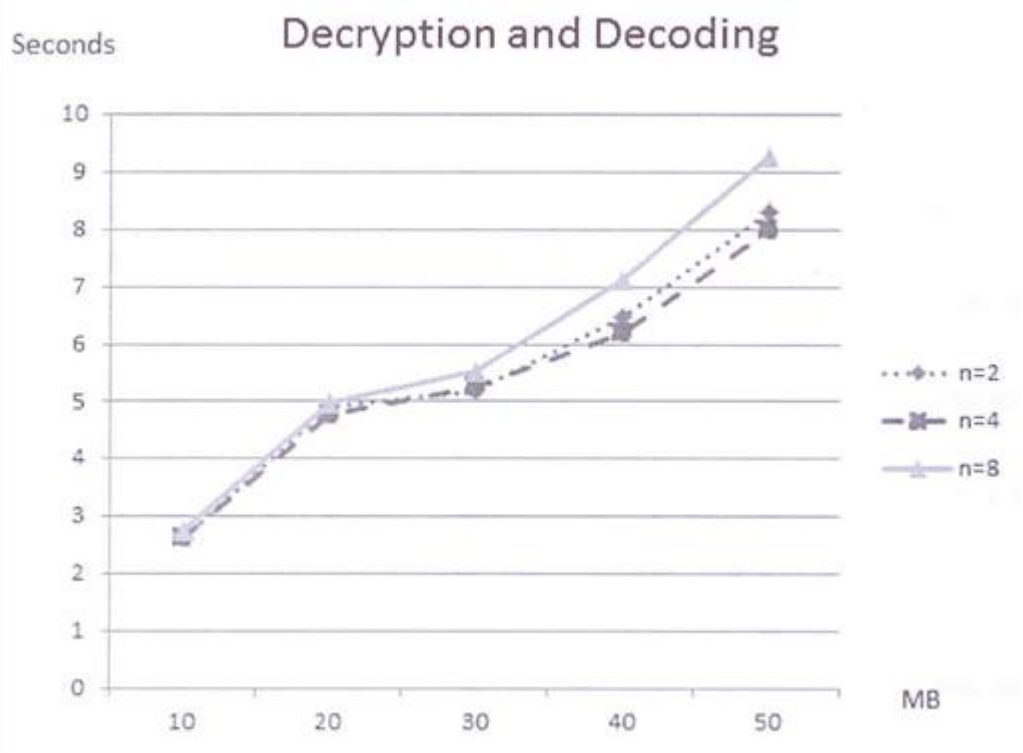
效率測量：

我們在 Intel Core i7 Q720 @ 1.60GHz with 4GB Ram 的機器上測試檔案處理的速度，下面的圖表分別顯示：

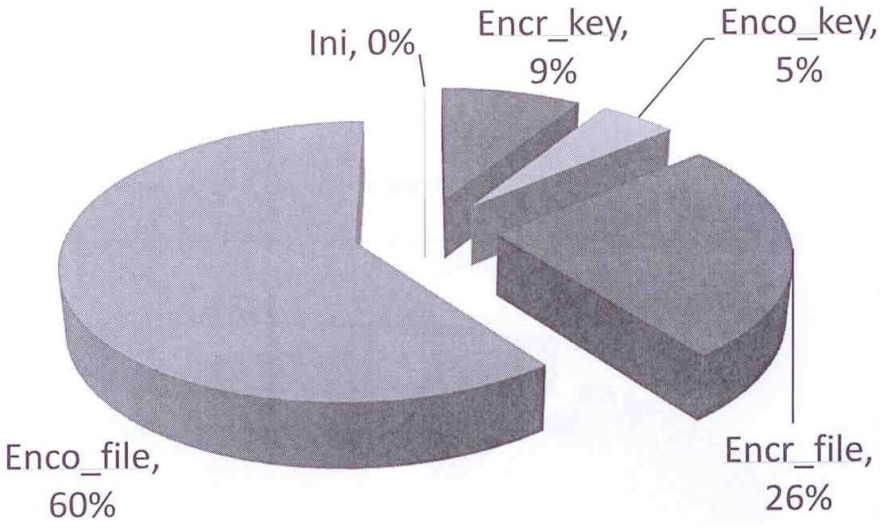
- 不同的檔案大小以及不同的 Codeword 數目花費的加密編碼時間



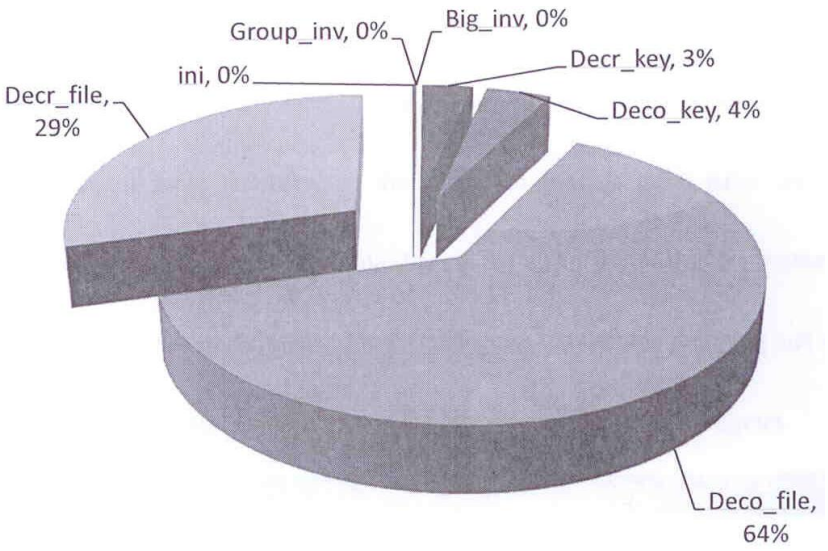
- 不同的檔案大小以及不同的 Codeword 數目花費的解密解碼時間



- 加密編碼時每個動作的花費比例



- 解密解碼時每個動作的花費比例

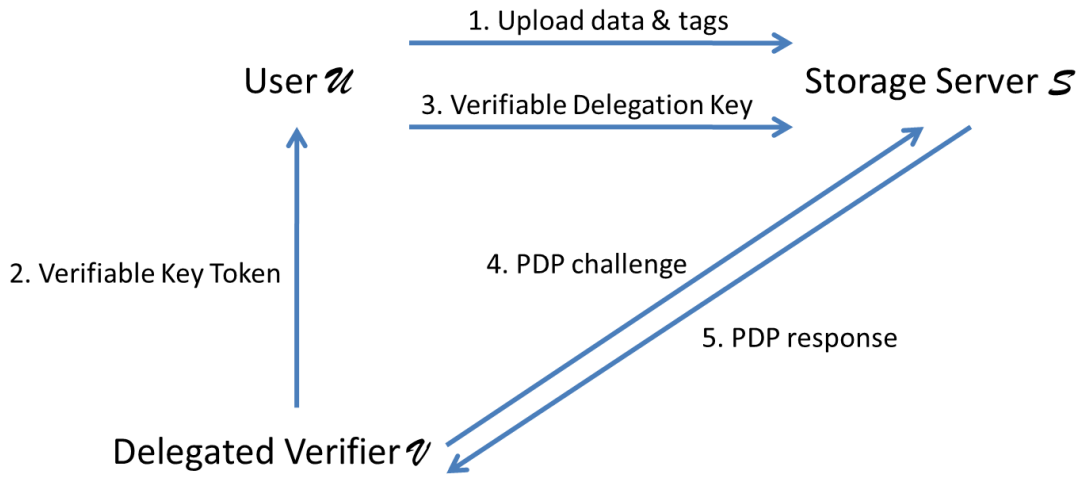


□ **資料完整性授權驗證機制**

在我們的模型中，總共有三個角色，使用者U（資料擁有者），被授權者V，以及儲存伺服器S。資料在上傳給儲存伺服器之前，使用者會先使用自己的私密金鑰產生資料對應的 tag，當要進行授權動作時，被授權者會利用自己的私密金鑰產生可驗證的 key token，並透過安全通道傳送 key token 給使用者，使用者利用被授權者的公開金鑰認證此 key token，通過驗證後再利用自己的私密金鑰及被授權者的 key token 產生可驗證的 delegation key，並傳送 delegation key 給儲存伺服器，儲存伺服器可以利用使用者和被授權者的公開金鑰去驗證 delegation key，通過驗證後授權動作即完成了，被授權者可以進行資料完整性驗證的動作。

進行資料完整性驗證時，使用者或是被授權人選擇欲驗證的資料區塊，並且產生對應的隨機 PDP challenge 給儲存伺服器，儲存伺服器收到 PDP challenge 後，利用所儲存的資料區塊以及 delegation key 產生相應的 PDP response。收到回傳的 PDP response 後，使用者或是被授權人利用自己的私密金鑰驗證此 response 的正確性，儲存伺服器只有在確實擁有資料區塊時才能產生正確的 PDP response。

在我們的模型設計中，被授權人無法再授權驗證能力給其他人，我們設計的原理是將關鍵的授權機制放置於 delegation key 中，有 delegation key 才能使得對應的被授權人獲得資料完整性驗證的能力，被授權人才能夠利用自己的私密金鑰驗證資料完整性。而在授權過程中，delegation key 是儲存於儲存伺服器中，被授權人沒有得到 delegation key 的資訊，既然被授權人被授權之後沒有得到新的資訊，被授權人的狀態和被授權之前的狀態一樣，被授權人當然不具有再授權的能力。



圖表三十三：授權驗證流程圖

我們提出的可授權資料完整性驗證機制包含了三個階段：設定階段，授權階段，以及驗證階段。設定階段包含了三個演算法：Setup，KeyGen，以及 TagGen。分別敘述如下：

- Setup(1^k)→π：由系統管理者執行，用來設定整個系統環境。管理者輸入安全參數 k，Setup 產生系統的公開參數 π。
- KeyGen(π)→(sk, pk, kt)：由使用者執行，用來產生使用者的金鑰組。使用者輸入系統公開參數 π，KeyGen 產生使用者的公開金鑰 pk，私密金鑰 sk，以及授權用的 key token kt。
- TagGen(π, sk, m)→(σ, t)：由使用者執行，用來產生資料的 tag 和 tag 的辨識碼。使用者輸入系統公開參數 π，私密金鑰 sk，以及使用者的資料 m，TagGen 產生可驗證 m 的 tag σ，以及辨識 tag σ 的識別碼 t。在實際建置中，識別碼 t 為使用者所挑選之字串。

授權階段包含了兩個演算法：GenDK 以及 VrfyDK。分別敘述如下：

- GenDK(π, sk_U, kt_V)→$dk_{U\to V}$：由使用者執行，用來產生對應於被授權人的 delegation key。使用者 U（授權人）收到被授權人 V 的 key token kt_V 後，輸入系統參數 π，自己的私密金鑰 sk_U，以及被授權人 V 的 key token kt_V，GenDK 產生對應於 U 授權給 V 的 delegation key $dk_{U\to V}$。
- VrfyDK(π, $dk_{U\to V}$, pk_U, pk_V)→{true, false}：由儲存伺服器執行，用來驗證收到的 delegation key 是否正確。儲存伺服器收到使用者 U 傳來的 delegation key $dk_{U\to V}$ 後，輸入系統參數 π，收到的 delegation key $dk_{U\to V}$，使用者 U 的公開金鑰 pk_U，以及被授權人 V 的公開金鑰 pk_V，VrfyDK 驗證 delegation key $dk_{U\to V}$ 是否正確。

驗證階段包含三個演算法：GenChal，GenProof，以及 VrfyProof。分別敘述如下：

- GenChal(π, t)→chal：由使用者或是被授權人執行，用來產生資料完整性驗證的 chalenge。使用者或是被授權人輸入系統參數 π，以及要驗證的資料區塊識別碼 t，

GenChal 產生對應的 PDP challenge chal。

- GenProof(π, m, σ, $dk_{U\to V}$, chal)→$pf_{chal, V}$：由儲存伺服器執行，用來產生資料完整性驗證的 response。儲存伺服器收到使用者U或是被授權人V的 PDP challenge 後，輸入系統參數 π，儲存的資料區塊 m，用來驗證 m 的 tag σ，delegation key $dk_{U\to U}$或是 delegation key $dk_{U\to V}$，以及收到的 PDP challenge chal，GenProof 產生對應的 PDP response $pf_{chal, U}$或是 PDP response $pf_{chal, V}$。
- VrfyProof(π, chal, $pf_{chal, V}$, t, sk_V)→{true, false}：由使用者或是被授權者執行，用來驗證儲存伺服器回傳的 PDP response 是否正確。使用者U或是被授權者V收到儲存伺服器回傳的 PDP response $pf_{chal, U}$或是 PDP response $pf_{chal, V}$後，輸入系統參數 π，PDP challenge chal，PDP response $pf_{chal, U}$或是 PDP response $pf_{chal, V}$，資料區塊 m 的識別碼 t，使用者U的私密金鑰 sk_U 或是被授權人V的私密金鑰 sk_V，VrfyProof 驗證 PDP response $pf_{chal, V}$ 是否正確。

我們的可授權資料完整應驗證方法如下所述：

- Setup：

  令 k 為安全參數，q 是一個 k-bit 的大質數，G=<g>和 G_T=<g_T>是 order-q 的雙線性乘法群，ê: G×G→G_T 是雙線性映射。系統管理者挑選三個密碼雜湊函數 H_1: {0, 1}*→G，H_2: G→G，H_3: $(Z_q)^*$→G。系統參數 π=(q, G, g, G_T, g_T, H_1, H_2, H_3)。
- KeyGen：

Key Generation. $\mathcal{U}$ chooses $x \in_R \mathbb{Z}_q$ as his private key $sk_{\mathcal{U}}$ and computes g^x as his public key $pk_{\mathcal{U}}$ and $H_2(g^x)^x$ as his key token $kt_{\mathcal{U}}$. $\mathcal{U}$'s key tuple is $(sk_{\mathcal{U}}, pk_{\mathcal{U}}, kt_{\mathcal{U}}) = (x, g^x, H_2(g^x)^x)$. $\mathcal{U}$ registers $pk_{\mathcal{U}}$ to the system manager.

- TagGen：

Tag Computation. $\mathcal{U}$ has data $\mathcal{M} = (m_1, m_2, \ldots, m_n)$, each block k-bit long, and would like to store them in $\mathcal{S}$. $\mathcal{U}$ chooses data identifier $h_{\mathcal{M}} \in_R G$ and tag identifier seed $T_{\mathcal{M}} \in_R \{0,1\}^*$ for $\mathcal{M}$. $\mathcal{U}$ may have many different data. Thus, he needs to choose a unique data identifier for each of his data. Each block m_i is tagged to a homomorphic verifiable tag σ_i which is identified by $h_{\mathcal{M}}$ and tag identifier $T_{\mathcal{M}}||i$. $\mathcal{U}$ computes these homomorphic verifiable tags $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ for $\mathcal{M}$ as follows:

$$\sigma_i = \left[H_1(T_{\mathcal{M}}||i) h_{\mathcal{M}}^{m_i}\right]^{sk_{\mathcal{U}}} \quad \text{, for } 1 \le i \le n \ .$$

$\mathcal{U}$ uploads $(\mathcal{M}, h_{\mathcal{M}}, \Sigma)$ to $\mathcal{S}$, and holds $(h_{\mathcal{M}}, T_{\mathcal{M}})$ for identifying and verifying Σ.

- GenDK & VrfyDK：

Delegation. $\mathcal{V}$ gives his key token $kt_{\mathcal{V}}$ to $\mathcal{U}$ over a secure channel, and obtains $h_{\mathcal{M}}$ and $T_{\mathcal{M}}$ from $\mathcal{U}$. $\mathcal{U}$ uses $\mathcal{V}$'s public key $pk_{\mathcal{V}}$ to verify validity of $kt_{\mathcal{V}}$ by checking whether $\hat{e}(g, kt_{\mathcal{V}}) = \hat{e}(pk_{\mathcal{V}}, H_2(pk_{\mathcal{V}}))$. Then, $\mathcal{U}$ computes the delegation key

$$dk_{\mathcal{U}\to\mathcal{V}} = kt_{\mathcal{V}}^{1/sk_{\mathcal{U}}}$$

and gives it to $\mathcal{S}$. $\mathcal{S}$ uses $pk_{\mathcal{U}}$ and $pk_{\mathcal{V}}$ to verify validity of $dk_{\mathcal{U}\to\mathcal{V}}$ by checking whether $\hat{e}(pk_{\mathcal{U}}, dk_{\mathcal{U}\to\mathcal{V}}) = \hat{e}(pk_{\mathcal{V}}, H_2(pk_{\mathcal{V}}))$. To revoke $\mathcal{V}$, $\mathcal{U}$ commands $\mathcal{S}$ to remove $dk_{\mathcal{U}\to\mathcal{V}}$ from its storage directly.

- GenChal & GenProof & VrfyProof：

Integrity Check. To check integrity of $\mathcal{M}$, $\mathcal{V}$ chooses coefficients $C = (c_1, c_2, \ldots, c_n) \in_R \mathbb{Z}_q^n$ and gives $\mathcal{S}$ the challenge

$$chal = \left(C,\ C',\ C''\right) = \left(C,\ h_{\mathcal{M}}^{s},\ H_3(C)^s\right) \quad \text{, where } s \in_R \mathbb{Z}_q \ .$$

After receiving $chal$, $\mathcal{S}$ verifies it by checking whether $\hat{e}(C', H_3(C)) = \hat{e}(h_{\mathcal{M}}, C'')$. If so, $\mathcal{S}$ uses $(\mathcal{M}, \Sigma, dk_{\mathcal{U}\to\mathcal{V}}, chal)$ to generate a proof $pf_{chal,\mathcal{V}} = (\rho, V, V', V'', V''')$ and gives it to $\mathcal{V}$ as a response. $pf_{chal,\mathcal{V}}$ is computed as follows:

$$pf_{chal,\mathcal{V}} = \left(\hat{e}(\prod_{i=1}^{n}\sigma_i^{c_i},\ dk_{\mathcal{U}\to\mathcal{V}})^t,\ C'^{\sum_{i=1}^{n} c_i m_i},\ C''^{\sum_{i=1}^{n} c_i m_i}, H_2(pk_{\mathcal{V}})^t,\ g^t\right) \quad \text{, where } t \in_R \mathbb{Z}_q \ .$$

After receiving $pf_{chal,\mathcal{V}}$, $\mathcal{V}$ uses $(sk_{\mathcal{V}}, h_{\mathcal{M}}, T_{\mathcal{M}}, C, s)$ to verify $pf_{chal,\mathcal{V}}$ by checking whether

- $\rho^s = \hat{e}\left(\prod_{i=1}^{n} H_1(T_{\mathcal{M}}||i)^{sc_i} V,\ V''\right)^{sk_{\mathcal{V}}}$
- $\hat{e}\left(V,\ H_3(C)\right) = \hat{e}\left(h_{\mathcal{M}},\ V'\right)$
- $\hat{e}\left(V'',\ g\right) = \hat{e}\left(H_2(pk_{\mathcal{V}}),\ V'''\right)$

我們提出的可授權資料驗證方法經過正規的安全性證明，證明我們的方法是 Proof unforgeability，Proof indistinguishability，以及 Delegation key unforgeability。Proof unforgeability 代表著儲存伺服器若能產生正確的 PDP response，則儲存伺服器一定擁有正確的資料區塊。

Theorem 1. *If the truncated BDHE problem is $(t, \epsilon, 1)$-secure, the above scheme is $(t - q_1 t_1 - q_2 t_2 - q_T t_T - q_K t_K - 2t_{\overline{A}}, \frac{2^\ell}{2^{\ell-(q_1+q_T)q_T}} \frac{2^k}{2^k-1}\epsilon)$ proof unforgeable in the random oracle model, where hash functions H_1 and H_2 are modeled as random oracles $\mathcal{O}_{\mathsf{H}_1}$ and $\mathcal{O}_{\mathsf{H}_2}$, (q_1, q_2, q_T, q_K) are the numbers of times that an adversary queries $(\mathcal{O}_{\mathsf{H}_1}, \mathcal{O}_{\mathsf{H}_2}, \mathcal{O}_{\mathsf{Tag}}, \mathcal{O}_{\mathsf{DK}})$-oracles, (t_1, t_2, t_T, t_K) are the time used by $(\mathcal{O}_{\mathsf{H}_1}, \mathcal{O}_{\mathsf{H}_2}, \mathcal{O}_{\mathsf{Tag}}, \mathcal{O}_{\mathsf{DK}})$-oracles to respond an oracle query, $t_{\overline{A}}$ is the time used by the KEA1 extractor $\overline{A}$ to extract an exponent, k is the security parameter, and ℓ is the bit-length of a tag identifier seed.*

Proof indistinguishability 代表著未被授權的第三方，無法分辨判斷儲存伺服器回傳的 PDP response 是否是正確的，也就是未被授權的第三方無法驗證資料的完整性。

Theorem 2. *If the truncated decisional BDHE problem is $(t, \epsilon, 1)$-secure, the above scheme is $(t - q_1 t_1 - q_2 t_2 - q_P t_P, 2\epsilon)$ proof indistinguishable in the random oracle model, where hash functions H_1 and H_2 are modeled as random oracles $\mathcal{O}_{\mathsf{H}_1}$ and $\mathcal{O}_{\mathsf{H}_2}$, (q_1, q_2, q_P) are the numbers of times that an adversary queries $(\mathcal{O}_{\mathsf{H}_1}, \mathcal{O}_{\mathsf{H}_2}, \mathcal{O}_{\mathsf{Proof}})$-oracles, and (t_1, t_2, t_P) are the time used by $(\mathcal{O}_{\mathsf{H}_1}, \mathcal{O}_{\mathsf{H}_2}, \mathcal{O}_{\mathsf{Proof}})$-oracles to respond an oracle query.*

Delegation key unforgeability 代表未被授權的第三方無法偽造 delegation key 給自己，即使是跟被授權人共謀，也就是被授權人無法轉授權驗證能力給第三方。

Theorem 3. *If the InvCDH problem is (t, ϵ)-secure, the above scheme is $(t - q_2 t_2 - q_D t_D - q_C t_C, eq_C\epsilon)$ delegation key unforgeable in the random oracle model, where hash function H_2 is modeled as random oracle $\mathcal{O}_{\mathsf{H}_2}$, e is the Euler's number, (q_2, q_D, q_C) are the numbers of times that an adversary queries $(\mathcal{O}_{\mathsf{H}_2}, \mathcal{O}_{\mathsf{Dlg}}, \mathcal{O}_{\mathsf{Cor}})$-oracles, and (t_2, t_D, t_C) are the time used by $(\mathcal{O}_{\mathsf{H}_2}, \mathcal{O}_{\mathsf{Dlg}}, \mathcal{O}_{\mathsf{Cor}})$-oracles to respond an oracle query.*

- **基於機器碼之 Windows 惡意程式行為分析雲端平台**

以下將分別介紹此雛型系統之實際佈建狀況、系統架構、工作流程、系統實作與效能分析。可顯示本計畫除新興技術的研究之外，亦注重務實的系統開發可行性。

□ 實際佈建

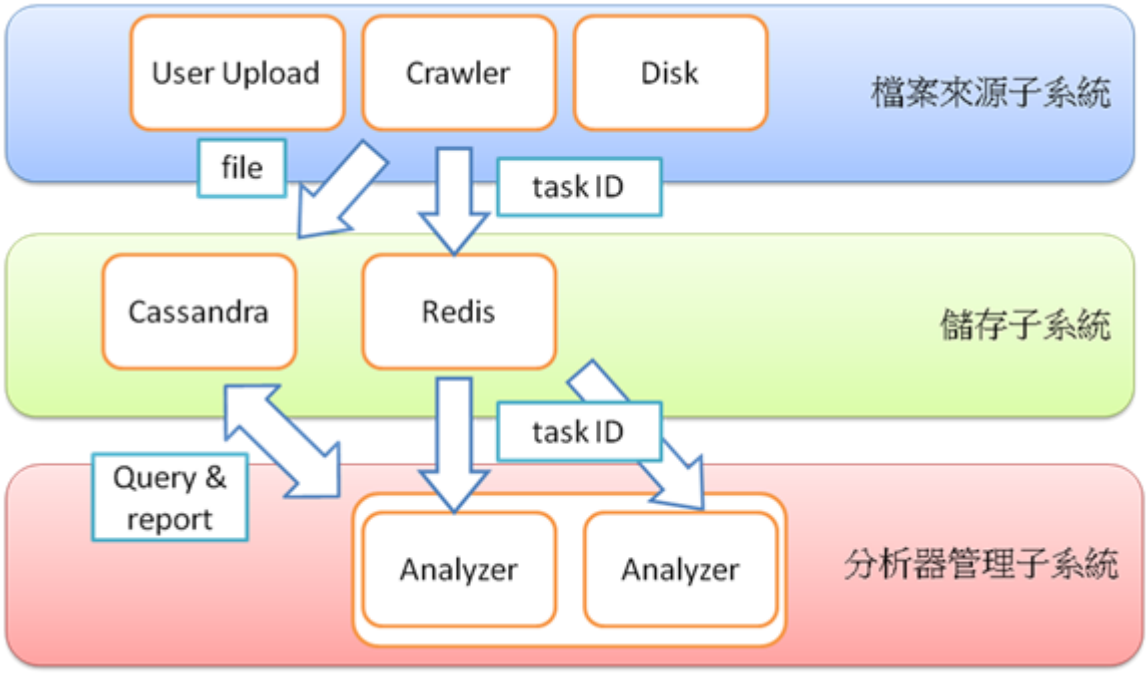
本計畫實際架設 Windows 惡意程式行為分析雲端平台，並建置於國立交通大學內，以進行雲端分析平台研究。本平台利用市售之高效能運算個人電腦，以有限的開發成本提供足夠的運算能量，並利用這些高運算量來實現「基於虛擬機器技術之動態程式行為分析」以及「高可靠性惡意程式樣本雲端儲存系統」。在動態程式行為分析部分，透過整合本實驗室所開發之惡意程式分析模組，有效地利用此平台的運算效能。而雲端樣本儲存系統，則可以大量的儲存已知或是分析過可疑的惡意樣本，以供往後的研究、查詢。此平台已經透過產學合作之關係，正與調查局、中華電信、趨勢科技、交通大學資訊服務中心等相關產業合作。進行實務性質的惡意程式分析，以提供多樣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

圖表三十四：WINDOWS 惡意程式分析雲端平台實際佈建

圖表三十五：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。



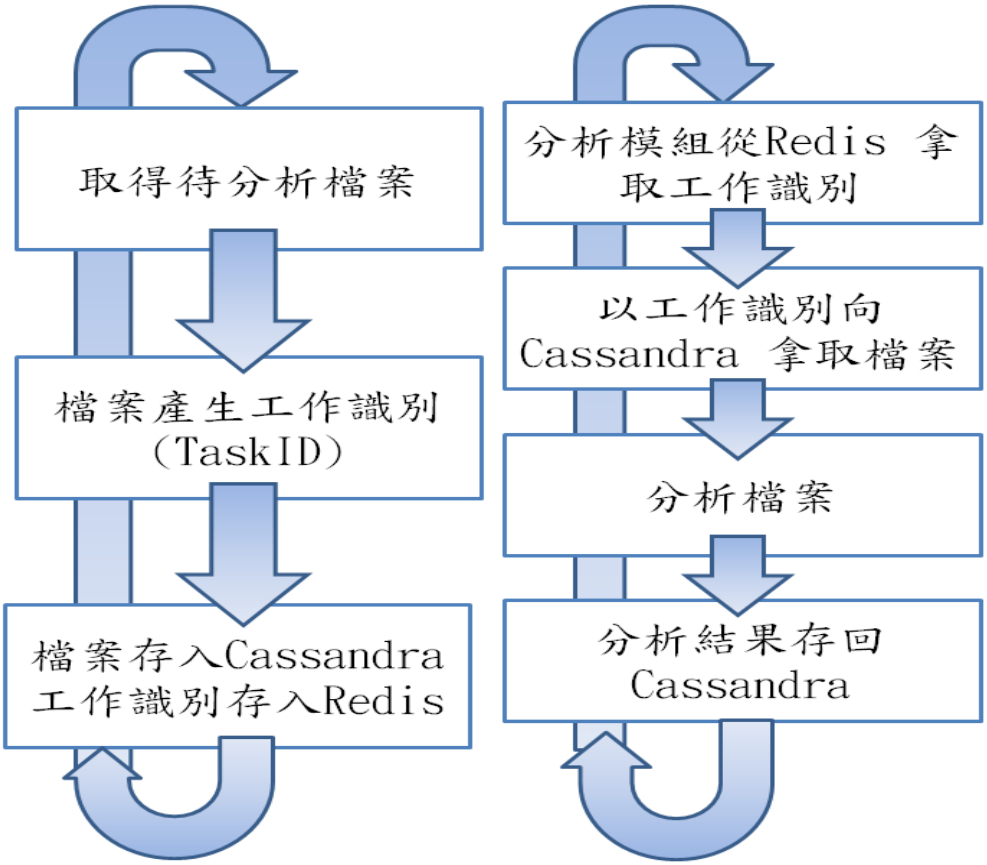
圖表三十六：基於機器碼之 Windows 惡意程式行為分析雲端平台系統架構圖

□ 工作流程

本系統的主要流程可分為兩種，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。



圖表三十七：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

以下將會列出實作的細節，並以其虛擬碼顯示各部分的流程。

程式 名稱	功能
secmap.rb	啟動本雲端惡意程式分析平台
startnutch.sh	啟動網路爬取子模組
NutchSlave.rb	傳送爬取網頁內容至分析平台
AnalyzerInvoker.rb	呼叫分析器子模組
common.rb	環境變數的設定和產生工作識別號碼（TaskID）
invokeAnalysis.rb	啟動分析器，之後拿取未分析檔案進行分析。
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。
putToCassandra.rb	Analysis input 元件將檔案放入 Cassandra。
getTaskID.rb	invokeAnalysis.rb 取得存放在 Redis 內的 TaskID。
getFileContent.rb	invokeAnalysis.rb 用 TaskID 至 Cassandra 取得相對應的檔案。
saveReportToCassandra.rb	分析器將分析結果存入 Cassandra。
getReportFromCassandra.rb	將分析器分析完的結果取出。

圖表三十八：虛擬碼列表

- 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();
}
```

圖表三十九：secmap.rb 之虛擬碼

本平台的運行主要下的指令如圖表三十八，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。./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。

說明如下:

- 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)
- 判別執行指令是’start’或是’stop’。
- 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**

圖表四十：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。

以下將對上圖做詳述:

(1) **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)。

(2) 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。

```
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/
```

(3) 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。

(4) 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。

(5) 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();
}
```

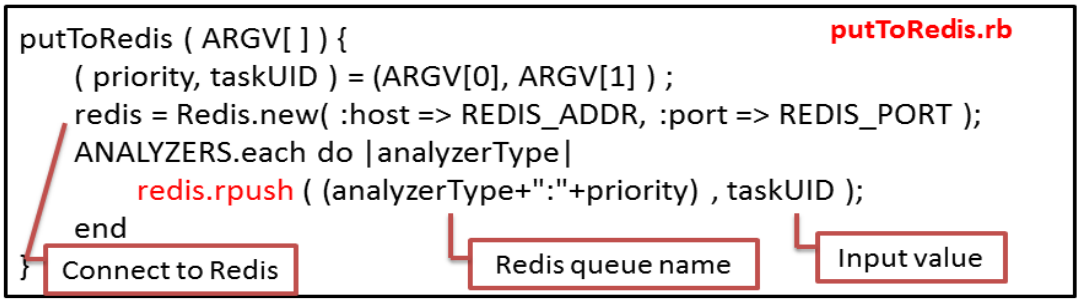
圖表四十一：NUTCH 之原始碼物件以及 NUTCHSLAVE.rb

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。

以下將做說明:

參數介紹：

- filename：Nutch 所抓取下來的檔案名稱。
- 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。
- 接著利用 loadCommandsTable()取得各個函式庫的位置。
- 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。
- 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。



圖表四十二：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。

以下將對圖表四十二做說明:

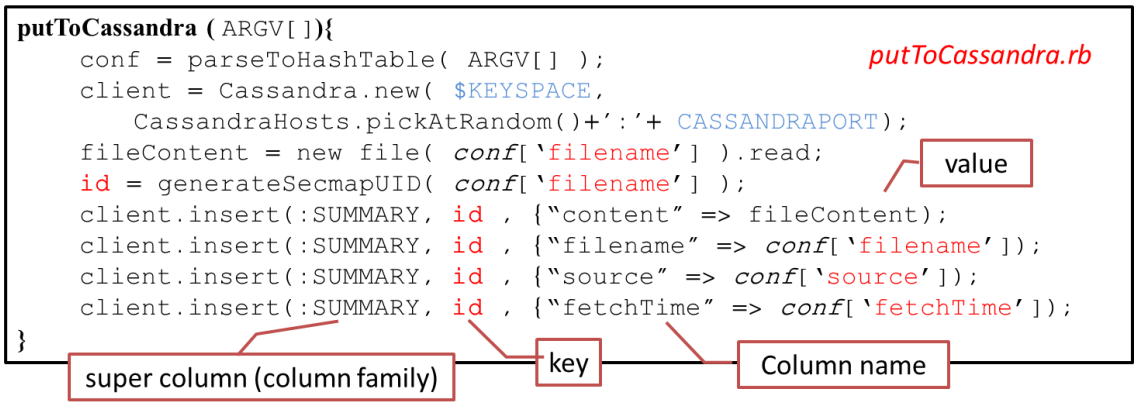
(一) 參數介紹:

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。



圖表四十三：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，如上圖所示:

（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();
    }
}
```

圖表四十四：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。以下將對上圖做詳述:

(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)
```

圖表四十五：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。

以下將對上圖做詳述:

(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.rb

```
getTaskID ( ARGV[] ){
    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;
}
```

圖表四十六：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。

(四) 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
```

圖表四十七：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[] ){
    client = Cassandra.new( KEYSPACE,                    getFileContent.rb
      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']);
}
```

圖表四十八：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:

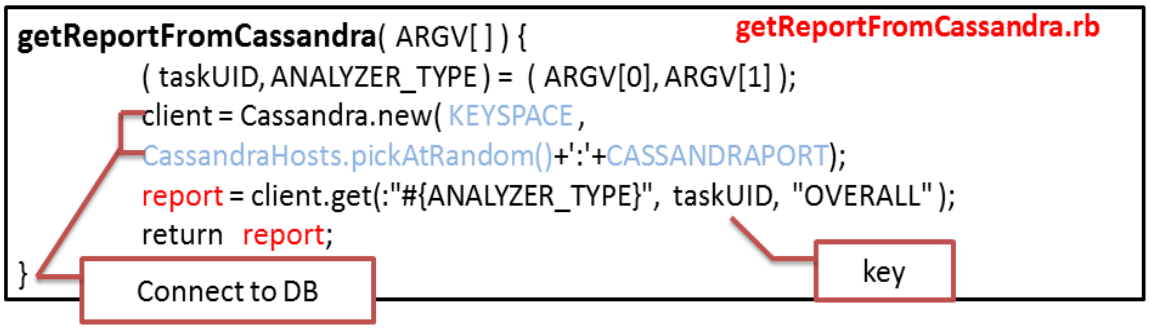
(一) 首先，連線至資料庫，其中 CassandraHosts.pickAtRandom()是用來連到不同的資料庫，回傳的值是任一資料庫的 IP 地址，目的是為了流量能夠均勻分到每一台分散式的資料庫上。

(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。



圖表四十九：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。

以下將對上圖做詳述:

(一) 參數介紹:

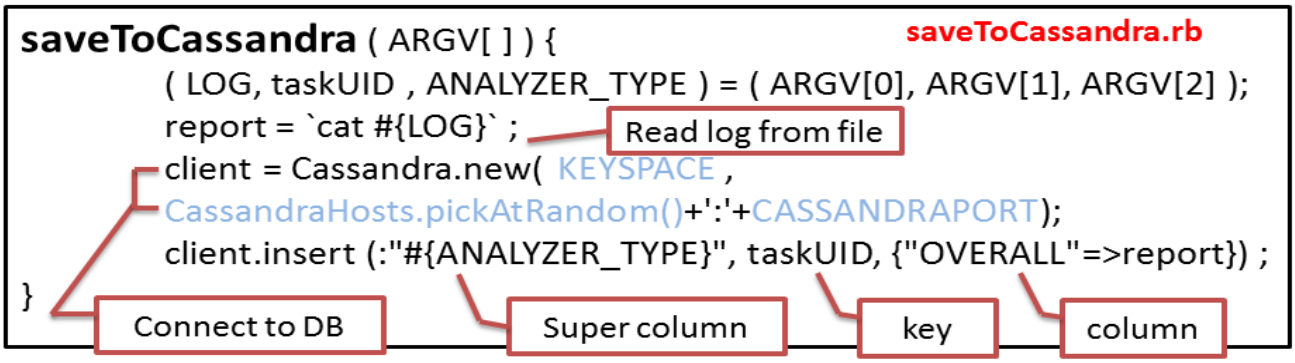
taskUID: 由分析檔案產生的 ID，產生方法則是將檔案內容作 MD5 以及 SHA1，並將兩者相連最後再連上檔案大小而得。

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。



圖表五十：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。

以下將對上圖做詳述:

(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。

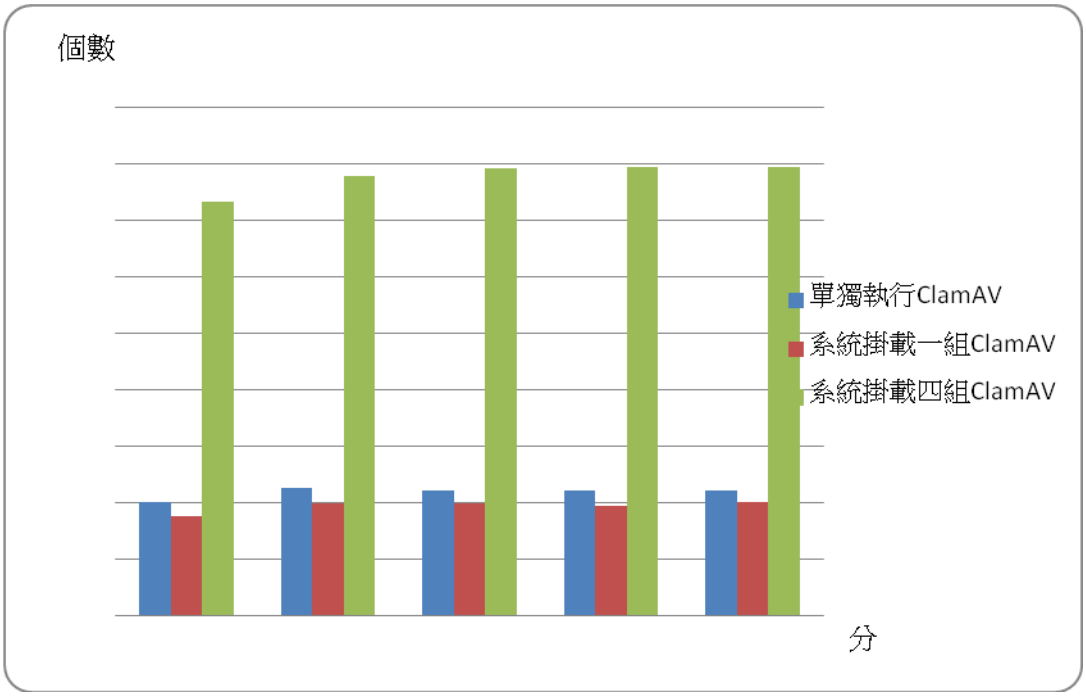
將報告存入資料庫中。透過 client.insert()來對資料庫的相對應欄位(super column:ANALYZER_TYPE 和 column name:OVERALL)填入相對的值(value)report 而 taskUID 則是 key。

□ 效能分析

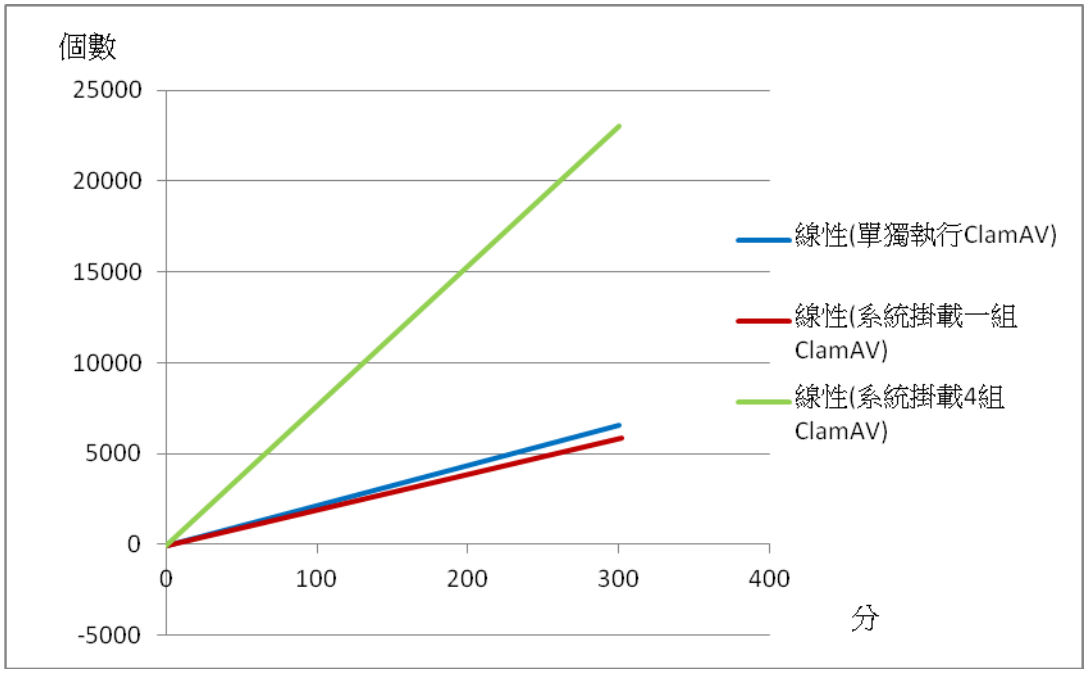
為了確認本系統的效能及可行性，我們設計了一套簡單的實驗。測試此系統在不同配置下實際運行的狀態，並與單獨分析器做比較。在此實驗中，系統採用的分析器皆 ClamAV。ClamAV 為一套開放原始碼的防毒軟體，用來偵測惡意軟體，普遍用於郵件伺服器。實驗所使用的機器皆使用 Intel(R) Core(TM) i7 CPU 搭配 16G 的記憶體。在此實驗中，本系統分別掛載執行一組 ClamAV 及四組 ClamAV，並與單獨執行 ClamAV 相比較。實驗分析目標為我們預先收集的惡意軟體樣本約 22000 隻，主要以 windows 的可能執行檔(exe)和動態連結函式庫為主(dll)。

如圖表五十一，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。

下圖為此實驗每 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，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。



圖表五十一：每 50 分鐘分析個數直方圖。



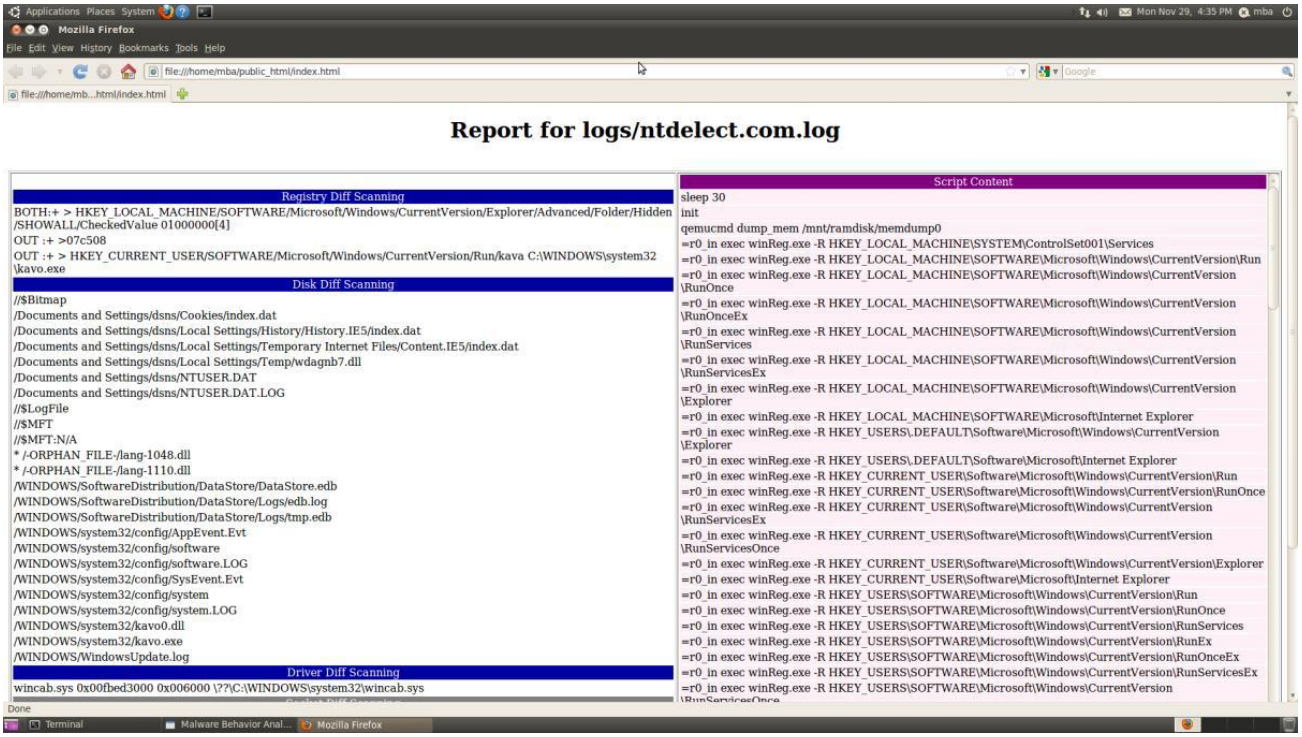
圖表五十二：時間-分析個數關係圖。

□ 運行截圖

圖表五十三：分析過程之紀錄檔案截圖

圖表五十四：放分析工做至惡意程式分析雲端平台

圖表五十五：網頁爬取子系統之紀錄截圖



圖表五十六：實際的惡意樣本分析報告

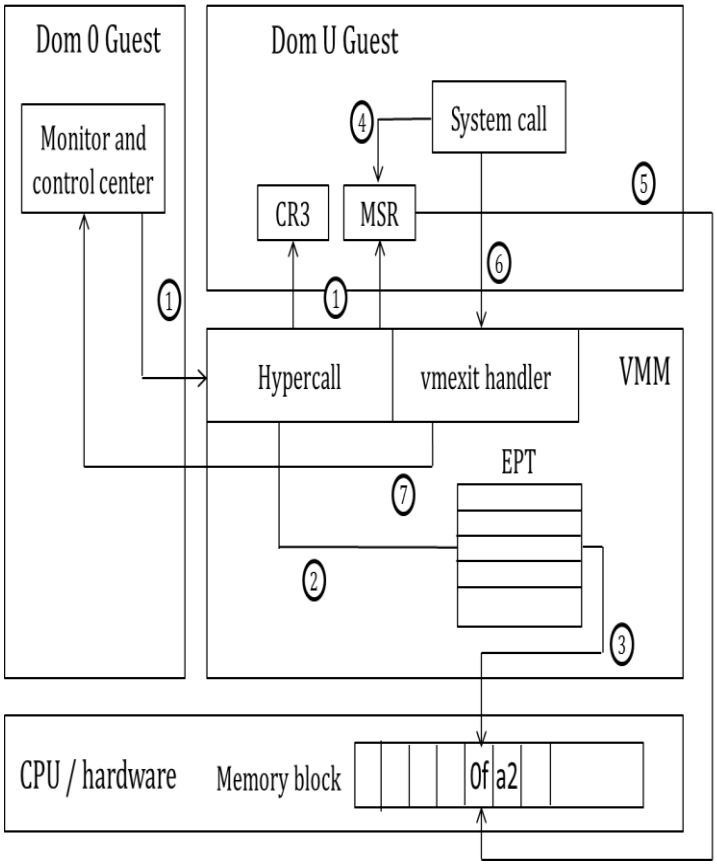
- **基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制(ICP)**

本系統將入侵偵測、入侵防護、反制的功能以及 ClamAV 整合進 Xen Hypervisor 中，總共分成四個部分：1.系統呼叫之攔截 2.記憶體內容物件之讀取、解析與制動 3. Online 式的檔案系統及磁區觀測、解析及制動技術 4.ClamAV 即時防護系統。另外在 Domain 0

Guest 中我們製作一個 Monitor & Control Center 來統籌該台實體機器上之虛擬環境整體的入侵偵測與反制工作。以下是每部份具體系統建置內容：

- **系統呼叫之攔截**

在系統呼叫這部分，我們提出的方式是使用特權指令(privileged instruction)來達到攔截的效果。特權指令是指 x86 架構下一些需要在 root mode 才能夠被執行的指令，因為這些指令會洩漏或是變更到系統底層硬體的資訊，這些資訊必須透過 Hypervisor 模擬，不能讓他們於硬體上直接執行，例如 cpuid 就是特權指令的一種(cpuid 會回傳 cpu 的相關硬體規格參數)。



Intercept system call

1) dom0 uses hypercall to get cr3 and MSR value of guest OS
2) VMM uses cr3 and MSR value to calculate the address of system call handler by looking up EPT
3) VMM writes the privileged instruction "cpuid:0fa2" into the address of system call handler
4) A system call causes Dom U guest to execute its system call handler
5) But Dom U actually executes the privileged instruction we inserted in the memory
6) It causes VMExit and is catched by the vmexit handler
7) VMM recognizes the system call and inform the Monitor and control center in Dom 0

圖表五十七：攔截系統呼叫流程圖

```
ENTRY(system_call)
        CFI_STARTPROC  simple
        CFI_SIGNAL_FRAME
        CFI_DEF_CFA    rsp,KERNEL_STACK_OFFSET
        CFI_REGISTER   rip,rcx
        /*CFI_REGISTER rflags,r11*/
        nop        <- 塞入兩個byte的空指令保留下來空間
        nop
        SWAPGS_UNSAFE_STACK
```

圖表五十八：linux/arch/x86/kernel/entry_64.S

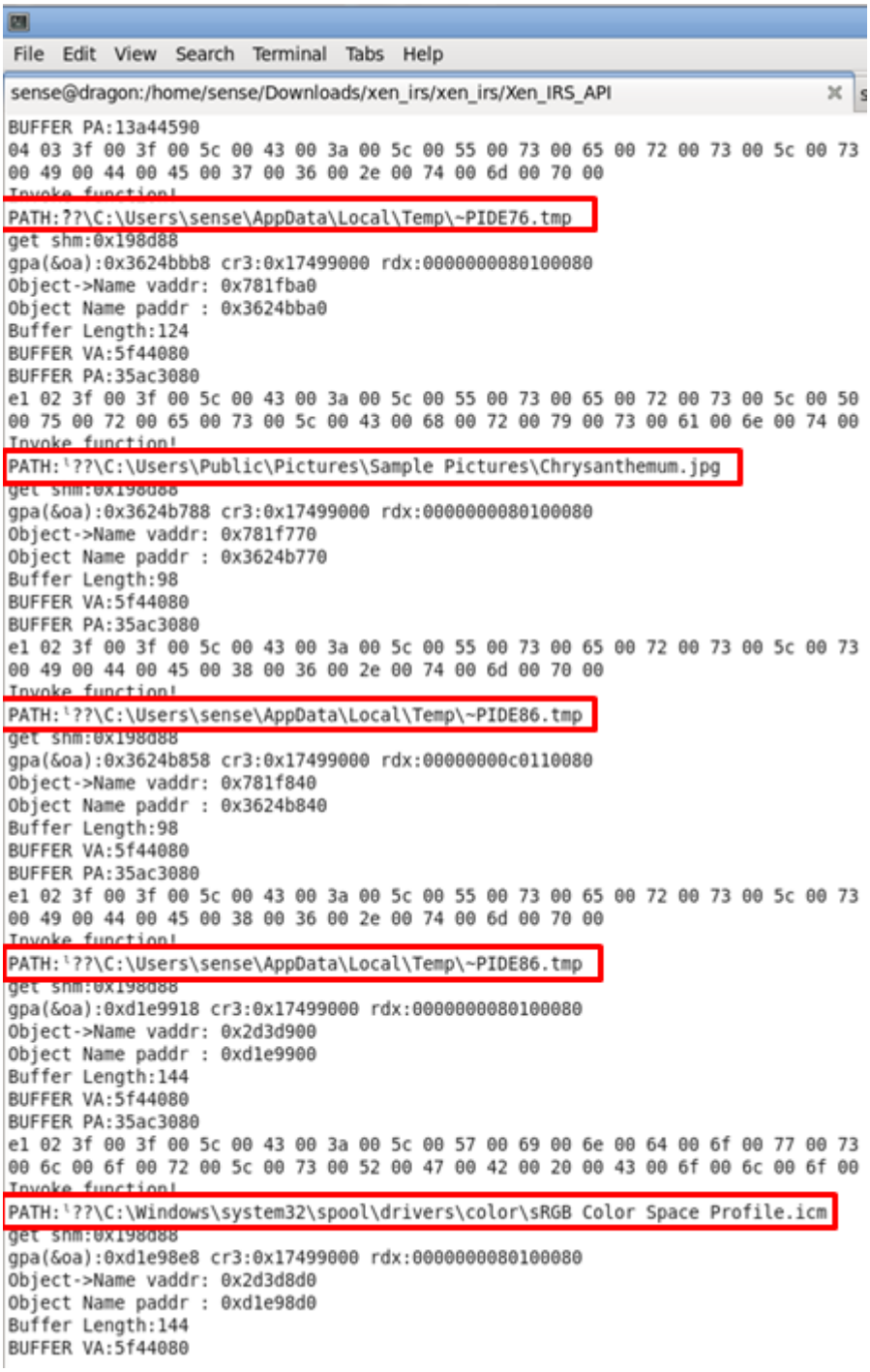
```
int insert_cpuid(struct domain *d, unsigned long gva)
{
        void *map_entry;
        if( (map_entry = map_specific_gva(d, gva)) == NULL)
                return -1;
        map_entry += (gva & 0xfff);
        atomic_set(&(d->arch.hvm_domain.is_monitored), 1);
        *(char*)map_entry = 0x0f;       ←插入特權指令
        *(char*)(map_entry+1) = 0xa2;  ←插入特權指令
        unmap_domain_page(map_entry);
        return 0;
}
```

圖表五十九：xen/arch/x86/mm/p2m.c

首先我們在 VMM 中 implement 一個新的 hypercall，使得 Dom 0 能夠透過 hypercall 介面來控制 Hypervisor，接著我們利用 hypercall 取得被監測者(Guest VM)的 CR3(control register 3)以及 MSR_LSTAR(存放系統呼叫處理函式的地址)的值(步驟 2)，並使用 EPT 來查詢所對應到的實體記憶體位置。最後將特權指令(cpuid)的 opcode(0fa2)覆寫到該記憶體位置中，不過這樣的動作可能會把原本存在於記憶體中的指令覆蓋掉而產生意外錯誤的情形，所以在這邊我們使用了一點小技巧：如果 Guest VM 裝載的是 Linux kernel，則我們重新編譯 Guest VM 的 kernel，使得 Guest VM kernel 內處理系統呼叫的函式多了兩個 byte 的空間；如果 Guest VM 裝載的是 Windows kernel，則我們在 Guest VM 的 kernel memory space 中尋找兩個 byte 的空間，這樣在填入特權指令的時候就不會影響到正常函式的執行。

```
asmlinkage void vmx_vmexit_handler(struct cpu_user_regs *regs)
{
unsigned int exit_reason, idtv_info, intr_info = 0, vector = 0;
unsigned long exit_qualification, inst_len = 0;
struct vcpu *v = current;
......
switch ( exit_reason )
{
        case EXIT_REASON_EXCEPTION_NMI:
        case EXIT_REASON_CPUID:
        inst_len = __get_instruction_length(); /* Safe: CPUID */
         __update_guest_eip(inst_len);
         vmx_do_cpuid(regs);
         break;
```

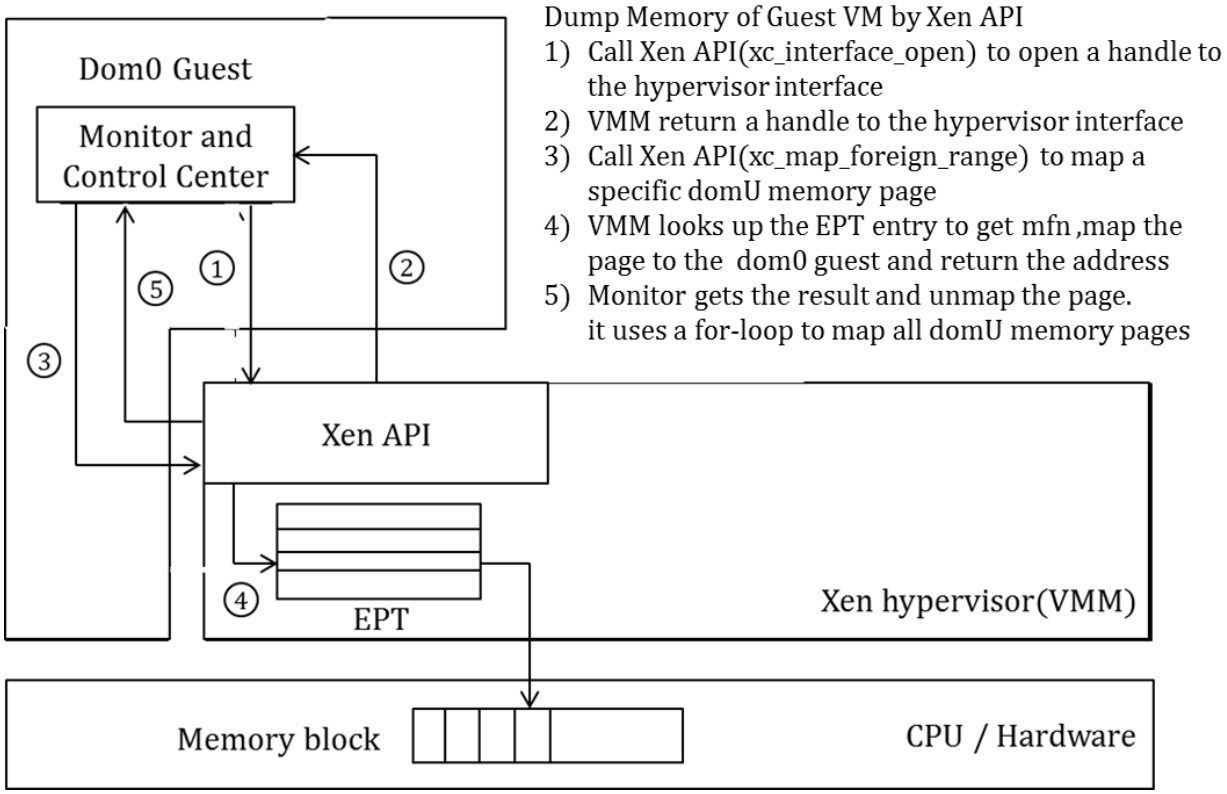
圖表六十：xen/arch/x86/hvm/vmx/vmx.c



圖表六十一：監控系統呼叫(NtOpenFile)執行結果

填入特權指令之後，一但 Guest VM 要執行系統呼叫，就會執行到我們所填入的特權指令(步驟 4)，而因為 Guest VM 不在 root mode，所以就會產生 VMExit 而進入位於 Xen 的 VMExit_handler。我們在 VMExit_handler 中會去判斷是否為系統呼叫所引起的 cpuid VMExit，如果這種情況，就會通知 Dom 0(步驟 7)，這樣就能達到監控攔截系統呼叫的功能。為攔截 NtOpenFile 的結果，紅色框框內為開啟的檔案路徑。

□ 記憶體內容物件之讀取、解析與制動



圖表六十二：記憶體內容物件讀取流程圖

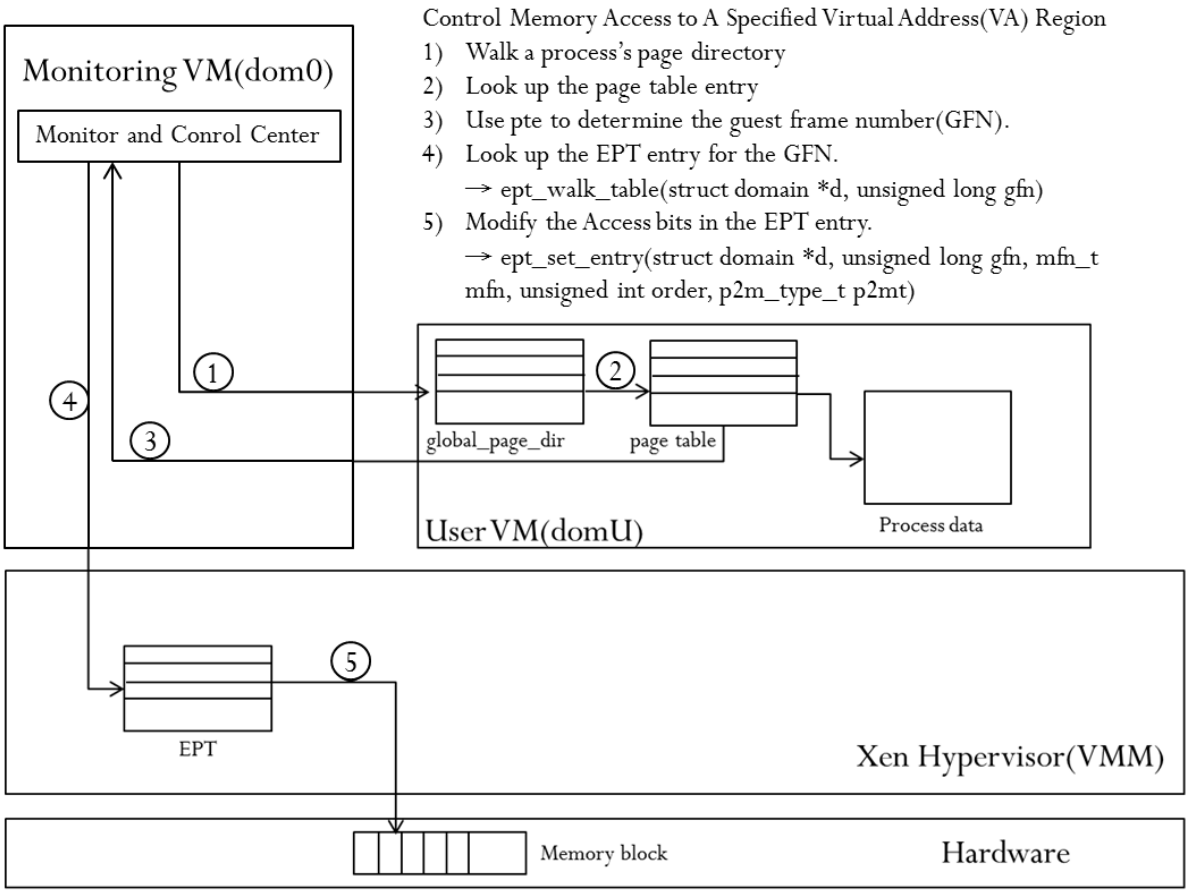
誠如先前所提到的，記憶體是 process 執行中不可或缺的資源，因此透過記憶體內容的讀取和解析來反制惡意的行為便顯得格外重要。在 Xen 的架構中 Guest VM 所使用的 memory space 必須經由 Xen Hypervisor 的對應，才能在實體記憶體佔有一塊空間，因此我們得先找出 Guest VM memory space 在實體記憶體上的地址，並根據此地址將資料讀取出來並進行解析。那在 Xen 的架構中我們要怎麼將 Guest_VM 的記憶體內容讀取出來呢？在步驟 1、步驟 2 及步驟 3 中，我們利用 Xen 提供映射 Guest VM 記憶體空間的 API (xc_map_foreign_range)，指定 Guest VM 的 domain id 以及目標 GPFN (guest physical frame number)給此 API 後，接下來的步驟 4 中 Hypervisor 便根據 EPT 將 GPFN 轉換成 MFN(machine frame number)，得到 MFN 後便可存取其中的資料。

圖表六十三：Guest VM 部分記憶體區塊內容

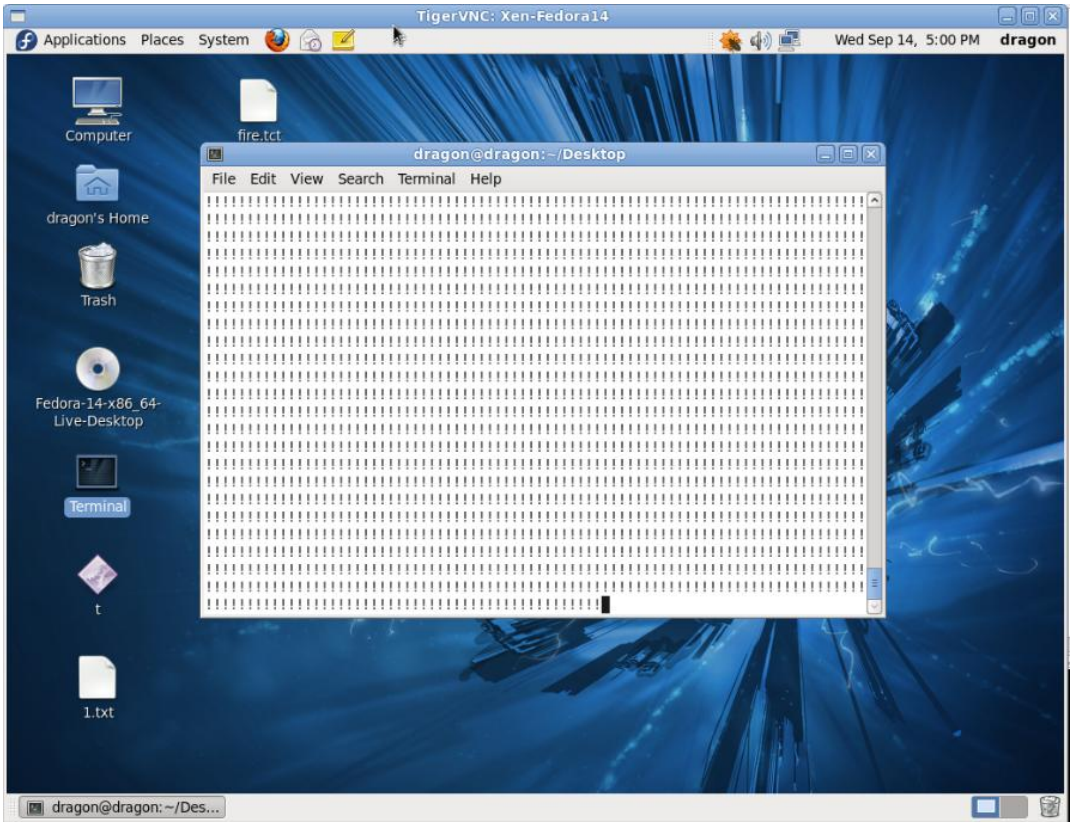
```
00000000  307830            xor [bx+si+0x30],bh
00000003  3130              xor [bx+si],si
00000005  353330            xor ax,0x3033
00000008  3030              xor [bx+si],dh
0000000A  2034              and [si],dh
0000000C  3938              cmp [bx+si],di
0000000E  64306330          xor [fs:bp+di+0x30],ah
00000012  636538            arpl [di+0x38],sp
00000015  396536            cmp [di+0x36],sp
00000018  3930              cmp [bx+si],si
0000001A  362030            and [ss:bx+si],dh
0000001D  3038              xor [bx+si],bh
0000001F  356330            xor ax,0x3063
00000022  3431              xor al,0x31
00000024  306639            xor [bp+0x39],ah
00000027  356335            xor ax,0x3563
0000002A  356120            xor ax,0x2061
0000002D  356234            xor ax,0x3462
00000030  3135              xor [di],si
00000032  6334              arpl [si],si
00000034  3438              xor al,0x38
```

圖表六十四：Guest VM 部分記憶體內容轉換結果

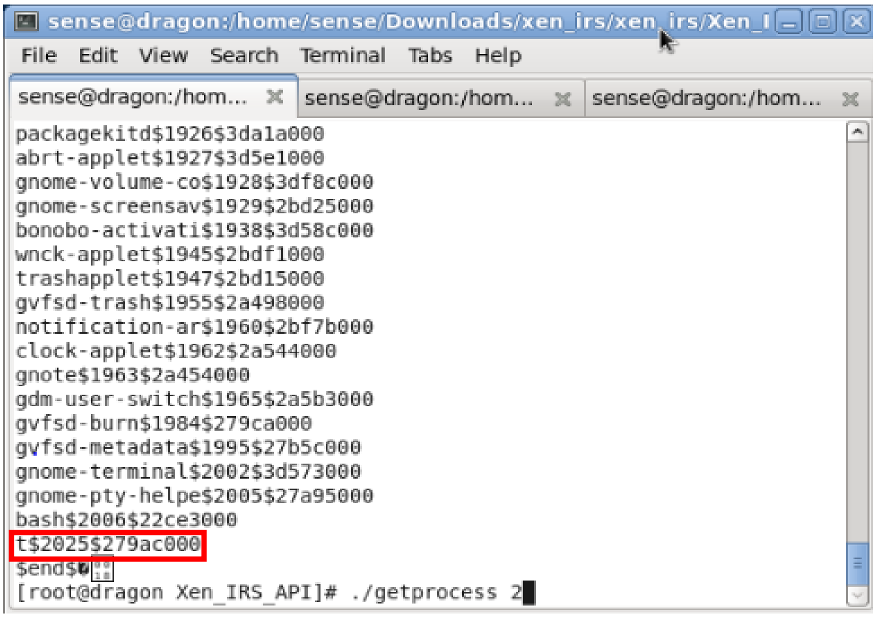
存取完記憶體後接下來的一大挑戰是對於所讀取到的記憶體區塊的內容必須要進行解析，memory 裡的資料是屬於 low level information，因此我們必須將 low level information 轉換成 high leve information，但在解析部分對於可疑的指令和程式流程轉移，我們還需要進一步開發相應的技術。另一方面，IDS/IPS 會需要對於記憶體空間有即時的監控能力。也就是說需要能即時掌握記憶體中的資料變化。



圖表六十五：利用 Xen API 進行 EPT Entry 權限管理

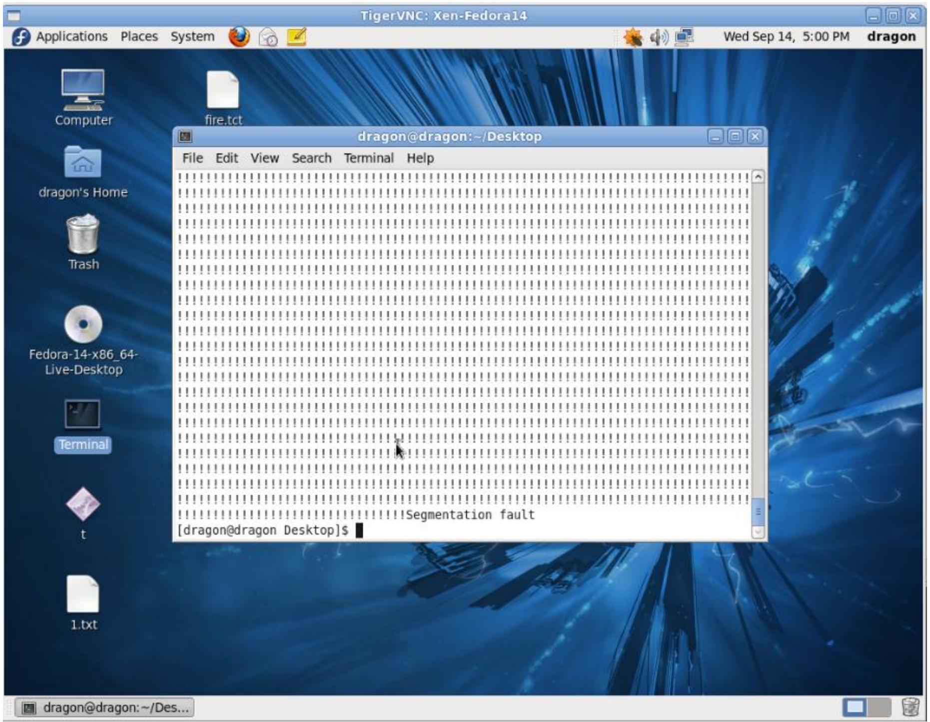


圖表六十六：未限制 Guest VM 存取 memroy block



圖表六十七：取得目標 process 資訊(namepidcr3)

圖表六十八：透過 guest pagetable 找到 process 所屬 gfn 資訊

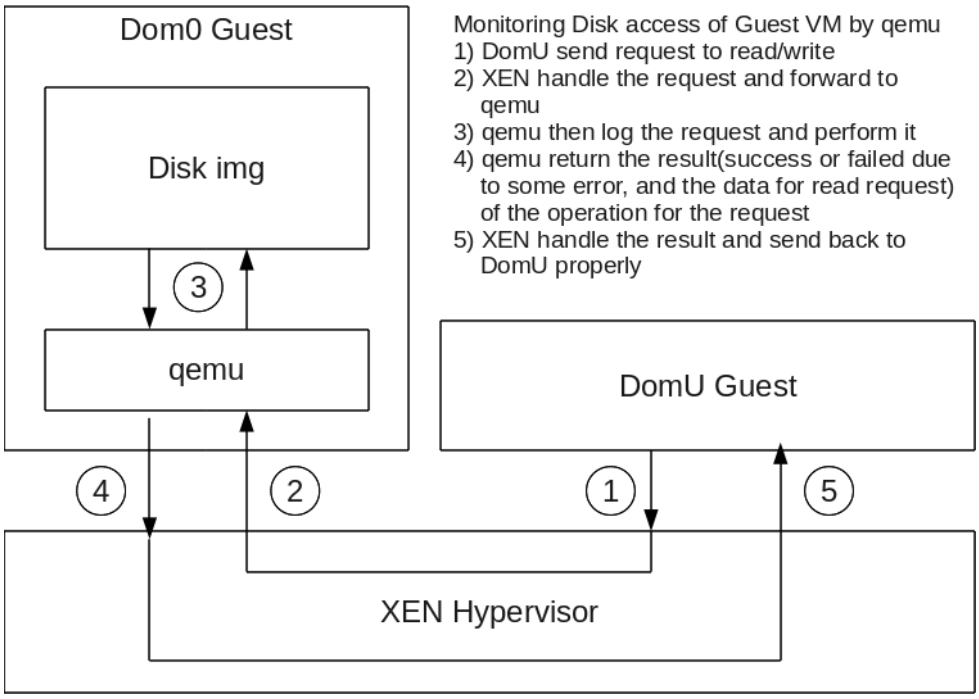


圖表六十九：限制 Guest VM 存取 memroy block

解析完記憶體區塊內容後，根據解析的結果來決定每個 memory page 在 EPT Entry 中的權限，透過權限管理的方式讓惡意軟體無法使用記憶體資源，藉此達到阻止惡意軟體的目的。接下來以圖表七十搭配實際範例操作截圖來進行說明，範例程式為一個

會不斷印出"!"，藉此佔用使用者 CPU 資源的惡意軟體，為了封鎖惡意軟體所擁有的記憶體資源，首先我們透過安裝在 Linux Guest VM 內的 driver 取得 process 的資訊，接著在步驟 1、步驟 2 及步驟 3 中我們透過 Guest VM 裡的 pagetable 找出惡意軟體所屬的 gfn，步驟 4 我們便利用 Xen API(ept_walk_table)找到惡意軟體在該 Guest VM 的 EPT 裡所對應到的 EPT Entry，步驟 5 透過修改 EPT Entry 權限的 API(ept_set_entry)將惡意軟體的 memory page 使用權限設定成 non-present(不能讀、寫和執行)，便可限制 Guest VM 裡惡意軟體對 memory page 的存取，藉此達到反制惡意軟體的效果。

□ 開發 online 式的檔案系統及磁區觀測及解析、制動技術



圖表七十：qemu 監控磁區流程圖

```
pthread_create(&sense_thread, NULL, &sense_loop, NULL);
main_loop();
pthread_join(sense_thread, NULL);
```

圖表七十一：tools/ioemu-qemu-xen/vl.c

```
    struct ntfs_device_operations ntfs_device_qemu_io_ops =
{
    .open           = ntfs_device_qemu_io_open,
    .close          = ntfs_device_qemu_io_close,
    .seek           = ntfs_device_qemu_io_seek,
    .read           = ntfs_device_qemu_io_read,
    .write          = ntfs_device_qemu_io_write,
    .pread          = ntfs_device_qemu_io_pread,
    .pwrite         = ntfs_device_qemu_io_pwrite,
    .sync           = ntfs_device_qemu_io_sync,
    .stat           = ntfs_device_qemu_io_stat,
    .ioctl          = ntfs_device_qemu_io_ioctl,
    };
```

圖表七十二：tools/ioemu-qemu-xen/fs-ntfs.c

```
    static struct struct_io_manager struct_qemu_manager = {
    EXT2_ET_MAGIC_IO_MANAGER,
    "QEMU I/O Manager",
    qemu_open,
    qemu_close,
    qemu_set_blksize,
    qemu_read_blk,
    qemu_write_blk,
    qemu_flush,
    qemu_write_byte,
    qemu_set_option,
    qemu_get_stats,
    qemu_read_blk64,
    qemu_write_blk64,
    qemu_discard,
    };
```

圖表七十三：tools/ioemu-qemu-xen/fs-ext2.c

在磁區觀測的部份，我們可以很清楚看到，從 Guest VM 來的磁區存取請求經由 Hypervisor 轉換後會送交給 Qemu 處理，因此所有的磁區操作最後都會經由 Qemu 來讀寫該系統的區塊裝置(像是映像檔或者是硬碟上的某一個分割區)。我們藉由修改 Xen 中 Qemu 的程式碼(位於 tools/ioemu-qemu-xen/)，在 qemu-dm 運行的主迴圈 main_loop 之外新增一個執行緒來做為監控中心與 Qemu 的介面聆聽及回覆監控中心的存取請求，檔案系統層級的存取部份，我們結合了 debug-fs 及 ntfs-3g 兩套函式庫來分別支援 ext 和 ntfs 檔案系統，在 Qemu 的程式碼加入了兩個檔案 fs-ext2.c 及 fs-ntfs.c，遵循該函式庫寫出底層 I/O 的對應函式，如此一來該函式庫即可透過這些函式直接使用 Qemu 中的 block I/O 函式存取底層的磁區。

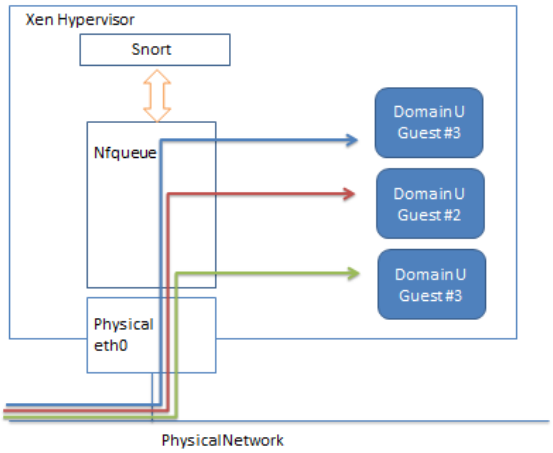
而關於制動的部分，我們使用 Qemu 將可疑檔案內容讀取出來做掃描，若該檔案含有病毒或其他惡意內容，我們便透過 Qemu 將該檔案刪除或將含有惡意內容的部份

移除)，或者我們也可針對某個檔案做動態監控，利用檔案系統的函式庫我們可得知該檔案所存放的磁區，之後若磁區有任何的更動都可以偵測到並加以過濾。

目前我們所實做的磁碟監控還無法從 Domain 0 中得知 Guest VM 裡的磁碟配置，許多作業系統會將磁碟分割成許多分割區，然後在作業系統運作時將分割區掛載起來形成完整的檔案系統配置，這部份的支援也許需要再搭配分割表的分析、開機程式設定及系統的檔案系統掛載設定分析等才能夠將這個功能做得更完整與自動化。

□ 網路觀測與制動技術

Snort 是一套開放原始碼的網路入侵預防軟體與網路入侵檢測軟體。Snort 使用了以偵測簽章（signature-based）與通訊協定的偵測方法。。Snort 可偵測惡意的網路攻擊封包，產生警告訊息並將其攔截。Snort 有三種封包監聽模式可以運行，分別是 sniff mode、packet log mode 和 inline mode。在 inline mode 下，Snort 會攔截住網路上的封包，在經過確認不是攻擊封包後才會將其放行至內部網路。在我們的系統中，將採用的是 inline mode。



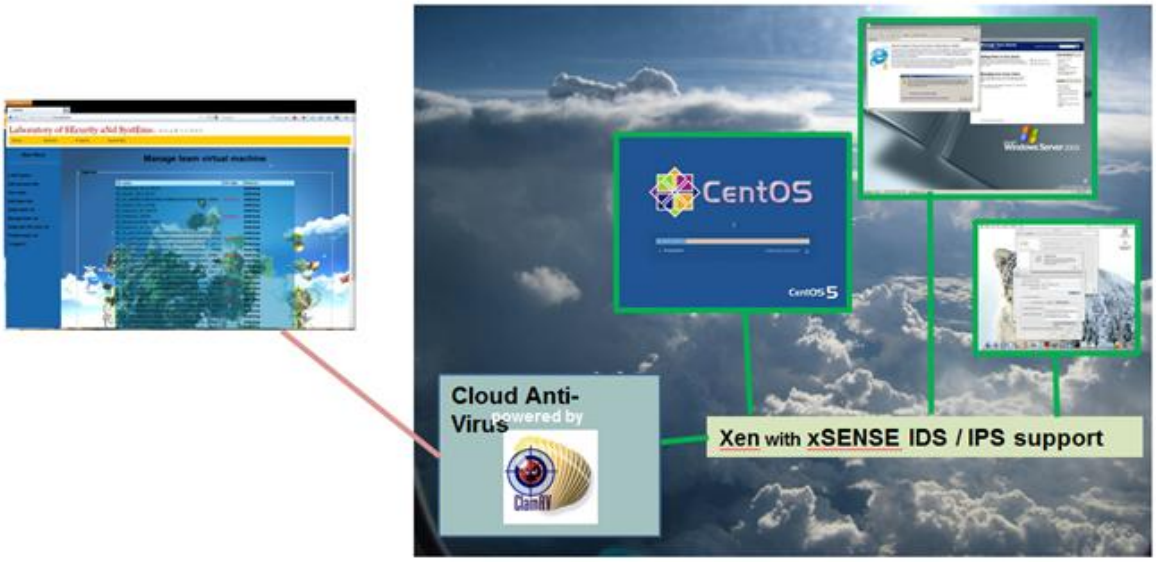
圖表七十四 Xen Hypervisor 結合 Snort 架構圖

上圖提到的是關於網路封包監測與監控的架構圖。首先，更改 Dom 0 的 kernel 相對應的地方，使得所有流經實體網路卡的封包都被導入 Nfqueue 中，也就是圖中分別要到三台 GuestVM 流量，在這邊用不同的顏色代表目的地不同的封包。接著 snort 會對所有進入 kernel nfqueue 中的封包進行檢測以及操作，可在 snort rules 中選擇當偵測到符合規則的封包時要 drop 或 alert，而我們將採用的系統可能會以先 drop 為考量。

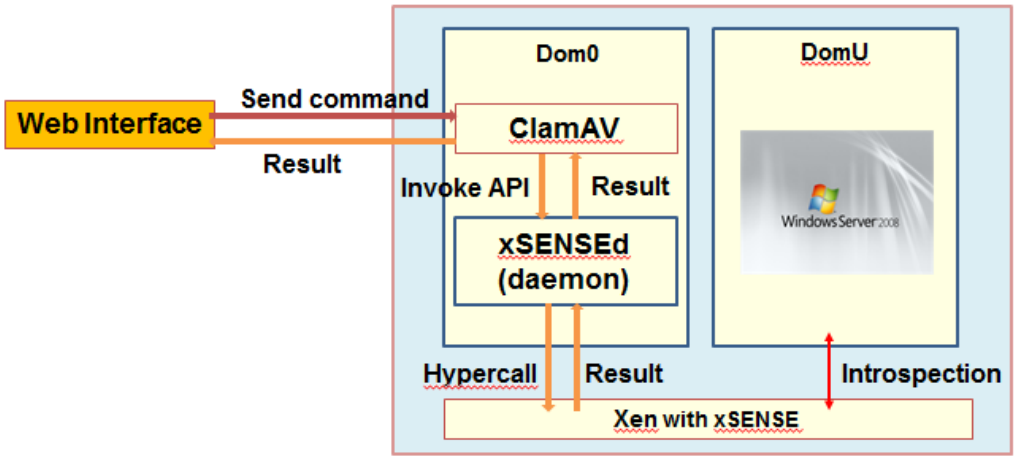
在網路安全部分，虛擬機是使用 bridge 的方式進行連接，透過 hypervisor 進行溝通。基於以上的機制，虛擬機網路和 hypervisor 是位於一樣的網路層，而非一種主從關係。如此一來，可以更改 hypervisor iptables 中的規則，使得虛擬機傳向到 hypervisor host 的封包都 dump 下來，杜絕任何虛擬機對於 host 有不正當的行為，並且分析起因和追蹤來向，在降低 hypervisor 受到攻擊的情形下同時以便進行對應的後續反制動作。

在阻擋對內的網路攻擊時，同時我們將透過一個反向 Network Intrusion Prevention System 來阻擋雲端平台上的攻擊實驗，可能在無意間影響到外部跨實體機器雲端網路或是純粹的外部網路的安全。並且，將在此設計用在各個區域的雲端網路之間，因此不同雲端間的網路或是雲端間的內部網路都有相對應的機制防禦與反制網路攻擊，以提供高安全性的實驗。

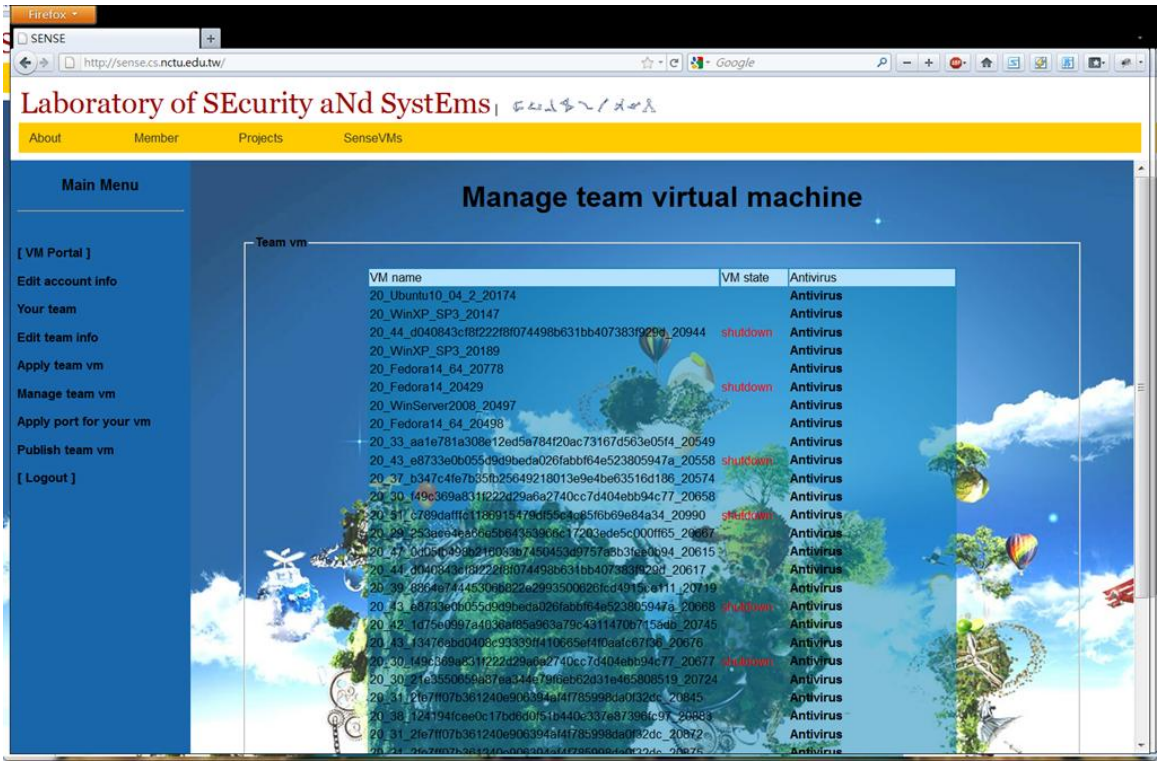
☐ ClamAV 即時防護系統



圖表七十五：ClamAV 即時防護系統概念圖

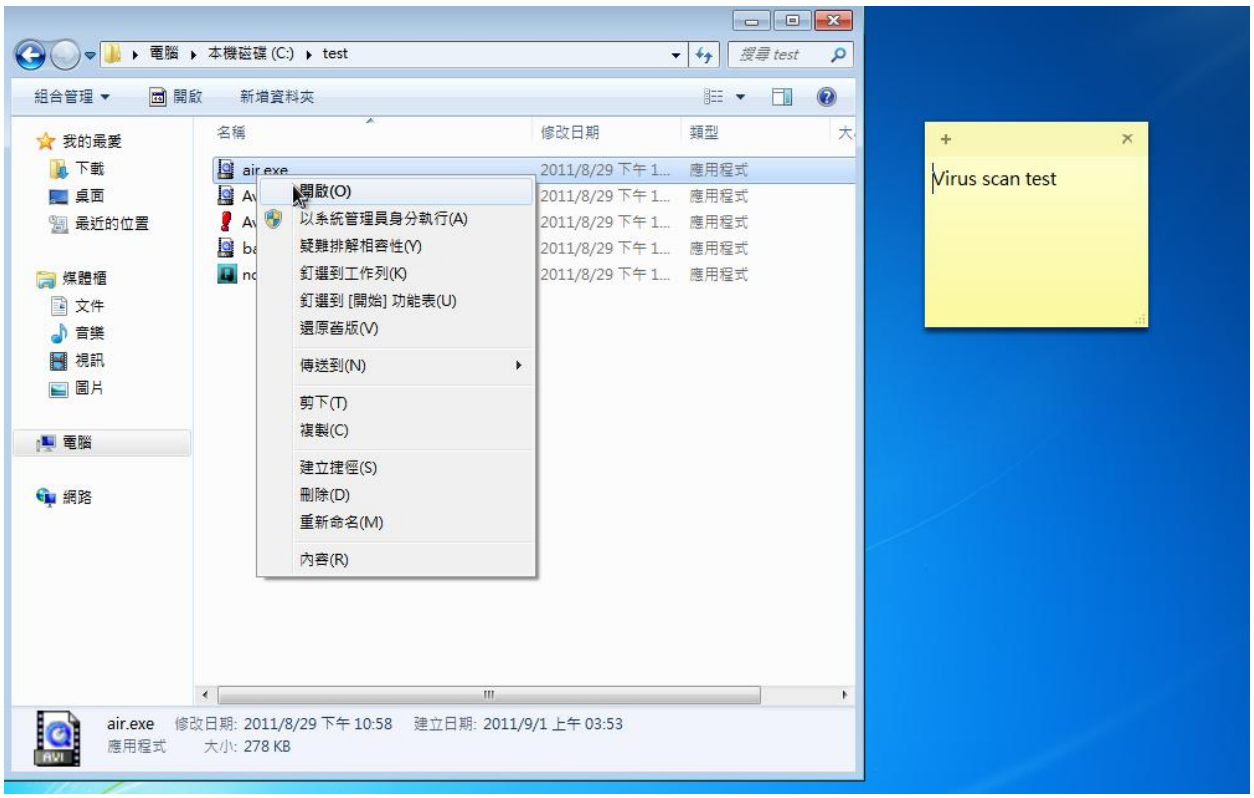


圖表七十六：ClamAV 即時防護系統架構圖

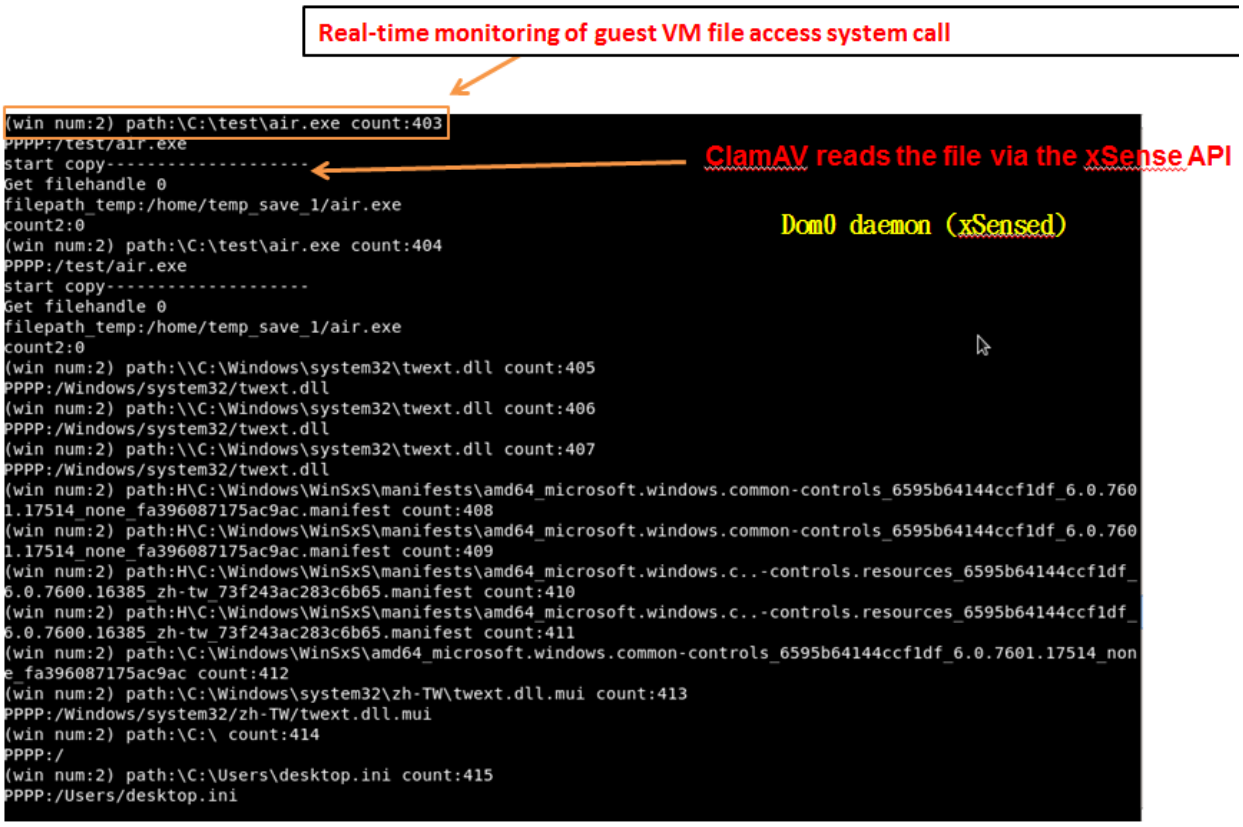


圖表七十七：前端 Web 操控介面

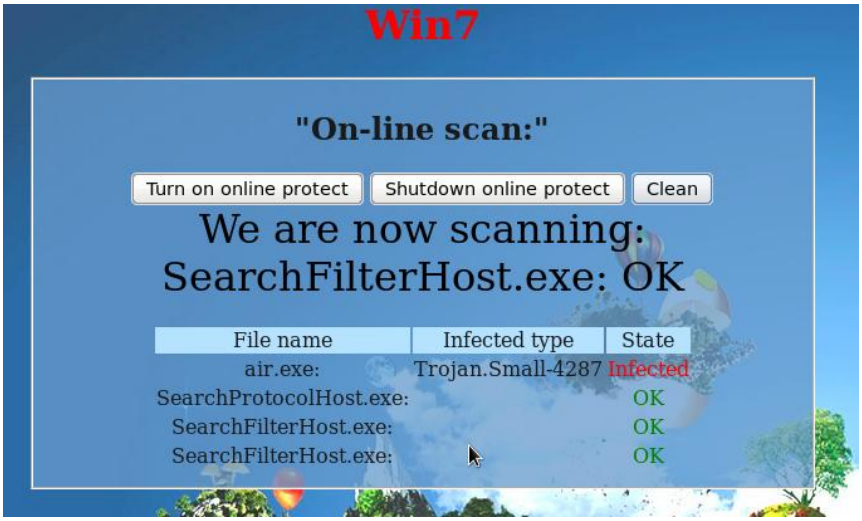
我們將攔截系統呼叫模組、磁碟檔案系統監控模組與 ClamAV 結合成一套 Xen 雲端環境入侵偵測與反制系統，管理者經由前端 Web Interface 對 ClamAV 下達指令後，ClamAV 透過我們定義好的 xSense API 介面將命令傳給 Monitor & Control Center(xSENSEd)，藉此來使用不同模組所提供的功能(攔截系統呼叫或監測磁碟檔案系統)，而 xSENSEd 收到 ClamAV 的命令後，便將命令透過 Xen Hypercall 介面傳進 Xen Hypervisor，再將 Xen Hypervisor 回傳的執行結果送回 ClamAV，最後透過 Web Interface 將結果顯示給管理者。



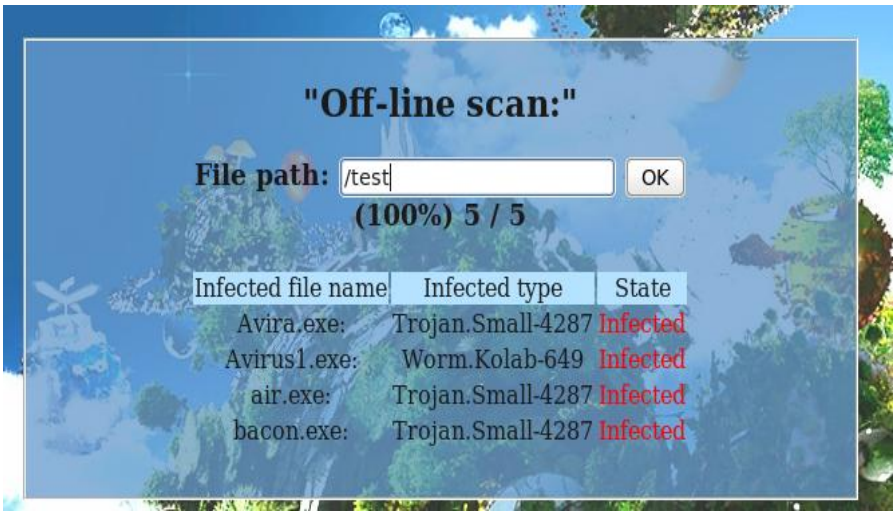
圖表七十八：Guest VM 使用者開啟病毒檔案



圖表七十九：ClamAV 存取 Guest VM 病毒檔案畫面



圖表八十：Online-Scan 掃描結果



圖表八十一：Offline-Scan 掃描結果

目前我們系統提供兩種防護模式，第一種模式為即時線上掃描(Online-Scan)，當使用者開啟 Guest VM 內的檔案時，控制中心會攔截系統呼叫(NtCreateFile & NtOpenFile)，並將檔案交由 ClamAV 進行掃描，掃描完成後結果會回傳至網頁介面上。第二種模式為離線掃描(Offline-Scan)，管理者可透過網頁介面將 Guest VM 下可疑檔案/資料夾路徑傳給 ClamAV 進行掃描，掃描結果亦會顯示在網頁上(掃描 C:\test 資料夾路徑下的所有檔案) 。

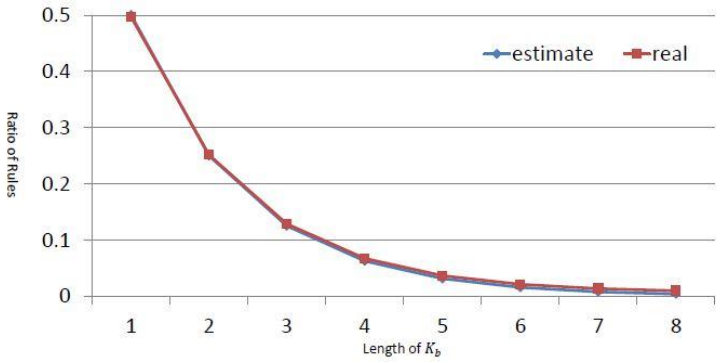
肆、 技術方案優越性

相較於現有之系統，本計畫所開發各項技術與雛型系統之技術方案優越性如下：

- **支援多樣功能之雲端資料安全儲存**
  - □ **保護隱私的雲端入侵偵測**

(1) 隱私性評估

我們針對單一關鍵字做隱私評估，其中，使用者上傳資訊的隱私和加密金鑰雜湊部份的長度有關，長度越長會讓候選者名單變小；反之長度越短會讓候選者名單變大，更能保護上傳資訊。實驗結果如下圖所示。



圖表八十二：單一關鍵字"content"之隱私評估

橫軸代表加密金鑰雜湊部分的長度分別為1～8bits，縱軸代表包含關鍵字"content"的 rule 在候選者名單中所占的比率，即

$$\frac{\text{候選者名單中包含關鍵字"content"的數目}}{\text{所有 rule 中包含關鍵字"content"的數目}}$$

我們可以藉由調整加密金鑰雜奏部分的長度來改變候選者名單的大小，來決定保護上傳資訊的隱私。

(2) 偵測率評估

我們使用 DARPA98 當作資料集來測試 SNORT 和我們所建立的實驗，DARPA98 是一個收集網路封包的集合，它使用區域網路去模擬出真實網路的環境去蒐集裡面一段時間內的封包，其中在這段時間內，它會產生一些假攻擊，散佈在這個網路環境中。DARPA98 可以被用來當作訓練的資料來訓練網路型的入侵偵測系統，再藉由分析網路封包來判斷此訓練的結果成效如何。

我們發現 SNORT 和我們建立的實驗所產生的警告數目一樣多，由此我們可知，藉由這個方法轉換 SNORT 規則並不會影響到原來的偵測能力。若要加強其偵測能力，則我們必須擁有更佳偵測率的 SNORT 規則，才有辦法增加偵測能力。

(3) 效能評估

我們在實驗中得知，採用保護隱私的方法所花費的時間和沒有保護所花費的時間影響不大，因此，採用此方法並不會對效能產生太大的影響。若要改善其效能，必需改變 Hadoop 的架構或是採用其他雲端環境建立的模型。

- **非集中式雲端儲存系統中系統修復機制**

我們將設計的修復機制與四個研究學者已經提出的修復機制進行比較，首先我們選取了 Regenerating code 中具有最小修復網路傳輸量的修復機制 MBR 以及具有最小儲存量的修復機制 MSR，另外我們也挑選了可以修復多個錯誤的修復機制 MCR 與 SRHC，我們比較修復每個錯誤的時候所需要的連線數量 u，可以修復的錯誤量，每修復一個錯誤所需要耗費的網路傳輸量，每個儲存伺服器的儲存量，以及每個新增伺服器所執行的修復程序是否對稱。修復程序如果是不對稱的如同 SRHC 的方法，那麼系統就需要一個中央控制單位來提供新增的伺服器不同的資訊以進行不同的修復程序，如此一來，這個修復機制就不適合非集中式儲存系統。比較的數據整理於下表中。

	u	server failures	bandwidth	storage	type
MBR [1]	$n-1$	single	$\frac{(2n-2)l}{(2n-k-1)k}$	$\frac{(2n-2)l}{(2n-k-1)k}$	symmetric
MSR [1]	$k+1$	single	$\frac{(n-1)l}{(n-k)k}$	$\frac{l}{k}$	symmetric
MCR [9]	$n-1$	multiple	$\frac{(n-1)l}{(n-k)k}$	$\frac{l}{k}$	symmetric
SRHC [10]	$< k$	multiple	$\frac{ul}{k}$	$\frac{l}{k}$	asymmetric
Our work	$< k$	multiple	$\frac{ul}{k}$	$\frac{l}{k}$	symmetric

圖表八十三：修復機制比較表

從表中可以看出，我們的方法可以為非集中式的儲存系統提供非集中式的修復機制，而且在修復的傳輸連線數量上可以低於 k，儲存量可以維持最佳值，但在傳輸量上有可能會比較多，尤其是在 k 值很小的時候。

- **資料完整性授權驗證機制**

我們的研究成果是第一個同時達到以下功能的資料完整性驗證方法：

(1) 有效率的資料完整性驗證
(2) 驗證能力的授權
(3) 被授權人再授權的防止
(4) 驗證能力的註銷

在效率方面的分析，我們分別針對各個演算法的計算量，各個角色的儲存量，以及各個階段的通訊量進行分析，分析結果呈列如下。授權階段的演算法具有效率，每授權一個人只需要常數次數的運算量。其他演算法則與別人的 PDP 方法一樣快。

Algorithm	Addition	Multiplication	Scalar Exponentiation	Hash	Pairing
Setup	0	0	0	0	0
KeyGen	0	0	2	1	0
TagGen	0	n	$2n$	n	0
GenDK	0	0	1	1	2
VrfyDK	0	0	0	1	2
GenChal	0	0	2	1	0
GenProof	$n-1$	$2n-1$	$n+5$	2	3
VrfyProof	0	n	$n+3$	$n+2$	5

- n is the number of data blocks

圖表八十四：Computation cost of each algorithm

使用者以及被授權者只需儲存常數般的儲存量，儲存伺服器則因為需要儲存資料區塊，所以需要跟資料量一樣級數的儲存量。

Party	Storage Cost (Bit)
User	$k+3p+\ell$
Delegated Verifier	$k+3p+\ell$
Storage Server	$nk+(1+n+v)p$

- k is the security parameter
- p is the size of an element in G
- l is the length of tag identifier seed $T_{\mathcal{M}}$
- n is the number of data blocks
- v is the number of delegated verifiers

圖表八十五：Computation cost

授權階段的通訊量也是常數大小，設定階段因為包含上傳資料的傳輸量，所以跟資料量相關，資料驗證階段因為需要傳送挑選哪些資料區塊做完整性檢驗，所以傳輸量最多也會跟資料區塊數量有關。如果使用隨機亂數產生器來決定哪些資料區塊要做檢驗，則只需要傳送亂數種子給儲存伺服器，這可以將資料驗證階段的傳輸量降低到常數大小。

Phase	Communication Cost (Bit)
Setup	$\mathcal{U} \rightarrow \mathcal{S} : nk+p+np$
Delegation	$\mathcal{V} \leftrightarrow \mathcal{U} : 2p+\ell$ $\mathcal{U} \rightarrow \mathcal{S} : p$
Integrity Check	$\mathcal{V} \leftrightarrow \mathcal{S} : nk+6p+p_T$

- k is the security parameter
- p is the size of an element in G
- p_T is the size of an element in G_T
- l is the length of tag identifier $h_{\mathcal{M}}$
- n is the number of data blocks

圖表八十六：Computation cost 2

- **基於機器碼之 Windows 惡意程式行為分析雲端平台**

雲端惡意程式分析平台之關鍵技術優勢可概分為以下三大方向：雲端節點機器管理、分析工作分配與排程、與可擴充式掛載介面，其比較如下表：

	傳統動態分析技術	**本系統**
管理系統支援種類	僅虛擬機器	**實體及虛擬機器**
管理系統容錯性	普通	**較高 (可處理斷網、斷電..等)**
分析排程技術	先進先出(FIFO)	**Priority Queue**
磁碟檔案讀寫加速	無	**利用記憶體加速**
整體執行效率	普通	**較高**
可擴充之分析模組	無	**有**
未來開發潛力	普通	**較高**

- **本系統可同時管理實體與虛擬機器，容錯性高於一般僅提供虛擬機器管理之系統**

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。

☐ **利用 Priority Queue 工作分配排程與記憶體儲存系統大幅改善分析速度**

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。

☐ **提供掛載介面以彈性擴充分析模組，相較於傳統封閉式設計有更高的開發潛力**

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。

- **基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制(ICP)**

	Ether	XenAccess	本平台
Xen 版本支援性	Xen3.1	Xen3.0~Xen3.3	Xen4.0
GuestVM 支援性	32 bits hvm guest	32 bits pv / hvm guest	64 bits hvm guest
Extended Page Table Support	×	×	✓
系統呼叫之攔測	✓	×	✓
記憶體內容物件之讀取、解析與制動	Δ（只監測是否對 memory 有寫入的動作，並無讀取、解析和制動記憶體內容物件功能）	Δ（只有讀取與解析記憶體內容物件，無制動功能）	✓

online 式的檔案系統及磁區觀測及解析、制動技術	×	×	✓

圖表八十七：本平台與其他 VMM 監測平台之比較

- **版本支援性**

(1) Xen 版本支援性

本平台支援 Xen4.0.版本，與其他 VMM 監測平台比較起來更為貼近 Xen 最新版本，而目前最新版本為 Xen4.1.1 版。

(2) Guest VM 版本支援性

本平台支援 64 bits hvm guest，而 64 bits guest 為未來不可避免的趨勢，因此比起其他 VMM 監測平台，本平台更加符合未來需求性。

- **功能性**

(1) 監測

本平台提供 Guest VM 系統呼叫、記憶體和磁碟系統監測，與其他監測平台比較起來功能更加完整。

(2) 制動

本平台提供 Guest VM 記憶體和磁碟系統制動功能，與其他監測平台比較起來功能不僅單純監測，同時能針對惡意軟體進行反制。

- **計與實作基於雲端技術之安全實驗觀測網路**

相較於其他實驗觀測平台，本系統具有以下三項特點:

- **虛擬機資源管理分析：**

將虛擬機技術導入網路測試平台，可依照各別測試的需求自動建立虛擬節點與其網路連結。

- **虛擬機安全性分析技術：**

此技術可確保虛擬節點之間的隔離性，防止運行中的惡意程式污染其他虛擬節點或是測試平台

- **虛擬機安全資源管理機制：**

此技術可建立自動化之虛擬機管理機制，測試平台可依照各別測試的需求管理各虛擬機之資源。

- **內建攻擊行為實驗：**

本計畫所建構 **vSWOON** 平台支援之攻擊行為實驗較為廣泛，其比較表如下：

攻擊行為	Emulab	DETER	TESTBED @TWISC	vSWOON
War Driving	Not Emphasized	N	N	Y
MAC Spoofing		N	N	Y
IP Spoofing		Y	Y	Y
Eavesdropping		Y	Y	Y
Wireless Eavesdropping		N	N	Y
Man-in-the-Middle		Y	Y	Y
Evil Twin		N	N	Y
DDoS		Y	Y	Y

伍、 結論與展望

本計畫已完成四大研究主題與多項雛型系統的開發，各研究主題除已獲得多項具體的研究成果以外亦具有高度的發展性。茲針對此四項研究主題之結論與展望說明如下：

- **支援多樣功能之雲端資料安全儲存**

  針對此研究主題我們完成了以下三項研究，其研究內容與未來之發展性說明如下：

  - **保護隱私的雲端入侵偵測**

    我們使用的 Song, Wanger, Perrig 的方法來保護使用者上傳到雲端的資料。這個保護隱私的能力取決於雲端提供者可以發現多少個候選者名單，這些候選者可能是使用者上傳的資料或不是。根據這個方法，我們利用 Hadoop 建造出雲端環境，轉換 SNORT 的規則檔當作我們的惡意特徵值比較資料庫，利用 jpcap 收集使用者端的封包以及 MapReuce 的模式撰寫特徵值比對的程式。藉由上述方法，我們建立了實驗的環境用來分析保護隱私的能力、偵測率和效能。透過實驗的結果，我們發現這個方法確實可以保護使用者上傳資料的隱私，而且也不會降低原來 SNORT 的偵測率。然而，將這個方法運作在 Hadoop 上無法得到即時的回應。未來我們將要改善這個實驗環境與雛型系統的實做細節，以確保系統的效能和穩定性，讓其能夠在實際的網路環境之中穩定地提供服務。

  - **非集中式雲端儲存系統中系統修復機制**

    針對非集中式的儲存系統，我們透過隨機程序的方法設計了一個非集中式的修復機制，我們的修復機制可以修復多個錯誤，並且在修復每個錯誤的時候使用較少的網路連線量。同時我們提出的機制維持了最佳的網路儲存量與合理的網路傳輸量。在研究的過程當中，我們注意到系統在修復多個錯誤的時候，可以批次的新增數個儲存伺服器，那麼這些新增的伺服器具有地理上的優勢可以進行低成本的雙向溝通，如果將這個實際上的狀況列入考量，那麼修復機制將可以設計得更有效率，這是我們未來努力的方向。另外要特別提出的議題是，為了瞭解具有機率的修復機制在實際系統中具體的成效如何，我們亦將可能進行的模擬實驗列入未來的研究計畫。

  - **資料完整性授權驗證機制**

    我們提出了一個可授權的資料完整性驗證模型，提供使用者資料的授權驗證功能。此模型允許受信任的第三方檢驗使用者資料的完整性，並且防止受信任的第三方將此驗證能力重新授權給他人。我們亦提供了一個可授權資料完整性驗證的方法，並正規證明此方法的安全性。目前此機制已可支援靜態資料的完整性驗證，但由於插入、刪除、修改之類的動態操作相當常見，如何支援有效率的動態操作將是我們未來的工作方向。

- **基於機器碼之 Windows 惡意程式行為分析雲端平台**

  雲端運算提供了新型態的運算架構以及提供龐大的資料儲存技術。藉著架設多台節點所形成的惡意程式分析雲端平台，可以有效地利用連結多台電腦的運算資源，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。

本平台也整合了網頁爬取子系統，該系統將會持續性的蒐集網路上可連結之資料，並且下載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，此方式可以更主動地去尋找可能受感染的網站，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。

目前此平台已整合本實驗室所開發之動態惡意程式分析模組。此分析技術因需要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。透過本計畫所開發之惡意程式分析雲端平台，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，惡意程式研究人員，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。經由在本報告內的效能分析驗證，利用多核心運算單元可以提供與預期相似的運算效能（理想值為 4 倍，實驗結果為 3.9 倍）。由此可證，本系統整合多台分析模組並不會產生過大的運算項能負荷。

在惡意樣本儲存方面，利用雲端儲存技術，可以提供常數時間的樣本存取。不同於以往傳統關聯式資料庫，可能會因為資料庫內容變大，而查詢的時間變長。同時，將單一個檔案分散儲存在不同機器內，可以方便做備份且加速讀取速度。目前已經蒐集有約兩萬隻的惡意程式樣本，並且儲存在惡意樣本資料庫內，並且皆有其動態程式行為分析報告。

除此之外，本計畫透過多方面的管道爭取到豐富的產學交流機會。此『基於機器碼之 Windows 惡意程式行為分析雲端平台』提供惡意樣本分析服務，正與法務部調查局、中華電信、趨勢科技、交通大學資訊服務中心等單位，做惡意樣本的知識交流。未來將持續透過與最前線的資安單位合作，讓本研究計畫可貼近真實的資安攻防事件。並透過合作交流擴大樣本蒐集並藉此調整分析引擎，以開發出更貼近目前實務需求的惡意程式分析系統。

- **與基於 Xen Hypervisor 之即時雲端環境入侵偵測與反制**

我們將自己開發出的 Xen Hypervisor 模組搭配 ClamAV 組合成一個基於 Xen Hypervisor 的即時雲端環境入侵偵測系統，本系統管理者可透過前端 Web Interface 來監控在 Xen Hypervisor 中運行的 Guest VM，而監控方式又分為兩種，第一種是線上即時監控，使用者在使用 Guest VM 時開啟的所有檔案都會即時的傳送給位於 Dom0 的 ClamAV 進行掃描，掃描結果亦會即時的顯示在網頁介面上。第二種是離線掃描，管理者可透過 Web Interface 將指定掃描的路徑傳給 ClamAV，ClamAV 將路徑下所有檔案掃描完成後，便會把掃描結果顯示在網頁介面上。

本系統雖能以 ClamAV 病毒引擎掃瞄出病毒檔案，但仍有許多可以改進的空間。以偵測的面向來說，目前僅提供系統呼叫、記憶體和檔案系統監測，但就使用現況而

言，網路已成為每位使用者不可缺少的工具，從網路發起的惡意攻擊也不在少數，因此如何從網路方面進行監測便是此系統所面臨的新課題。

以現有的系統來說，即時線上掃描功能目前僅能支援一個 HVM Guest VM，由於攔截系統呼叫後，當控制權從 Guest VM 轉至 Xen Hypervisor 時，搭載不同作業系統的 Guest VM 傳入 Xen Hypervisor 的參數格式完全相同，在無法辨別作業系統的情況下，便無法取得待掃描的檔案路徑位置，同時因為攔截系統呼叫模組會修改 Guest VM 的記憶體區塊內容，一但被 Windows 7 64bits 架構下的 Patch Guard 發現，便會造成 Guest VM 顯示藍屏錯誤訊息，除此之外在檔案系統讀取部分，由於作業系統通常會將磁碟分割成許多分割區，然後在作業系統運作時將分割區掛載起來形成完整的檔案系統配置，但本系統目前還無法從 Dom0 中得知 Guest VM 內磁碟配置，因此目前 Guest VM 內磁碟系統暫時只有一個磁區，這部份也許需要再搭配分割表的分析、開機程式設定及系統的檔案系統掛載設定分析，才能夠將這個功能做得更加完整，而在日後我們也會新增對 Paravirtualize Guest VM 的支援。

本系統雖然尚未將記憶體監測模組整合進來，但以記憶體為基礎的防護機制確實相當重要，不管是一個等待被執行的惡意程式、著名的 buffer overflow 或是一段被插入在 memory space 的 nop-slay malicious code，都可以藉由記憶體內容物件的讀取、解析與制動來加以防範，雖然目前在解析記憶體內容以及對於記憶體空間即時的監控能力我們還需要開發相關的技術，一旦我們掌握了此技術，對於 Xen Hypervisor 雲端環境下的入侵偵測與反制能力必定有相當的助益。

從效能面來看，本系統在開啟即時線上掃描後，根據掃描的檔案大小不同會造成不同的時間延遲(從開啟檔案到掃描完成所花的時間)，同時位於 Dom0 的控制中心在 Guest VM 密集觸發系統呼叫時，會占用 50~75%的 CPU 資源，以 server 等級的機器來看或許是小 case，但未來系統若擴充成能即時監控多個 Guest VM，每個 Guest VM 都需要 50~75%的 CPU 資源，即使是 server 級別的機器也吃不消，因此改善系統效能、增進系統穩定度和擴充系統功能便是我們未來持續努力的目標。

- **具仿真性與即時性的底層網路架構**

本計畫之研究成果將用於建置一個基於虛擬機之網路測試平台。此平台為一 hybrid 的網路實驗環境，同時提供虛擬主機與實體主機給使用者進行網路實驗。使用者可依照實驗的需求(例如；運算量、節點數量、頻寬、仿真度)，選擇使用虛擬主機或實體主機，或同時使用虛擬主機與實體主機進行實驗。因此，本團隊未來將朝下列數個領域繼續進行深入研究：

- **虛擬機於測試平台中的配置與規劃**

  本計畫將自動化虛擬機於測試平台中的佈署，並和現有實體主機的網路拓樸規劃介面進行整合，方便使用者配置實體主機與虛擬主機，減少建置虛擬化實驗環境的負擔。

- **虛擬機實驗環境之安全性與隔離性探討**

導入虛擬機技術之網路測試平台可能衍生相對的安全性問題，諸如：實驗隔離性(Isolation)、虛擬機安全漏洞防護等問題。本計畫將開發對應的安全防護機制，避免惡意的使用者藉由虛擬化技術之漏洞破壞網路測試平台。

- **虛擬機 QoS 管理**

  本計畫將利用今年產出的虛擬機效能評估報告，發展出一套配置虛擬化實驗節點的佈署機制，以確保每台虛擬節點擁有足夠的資源進行網路實驗，並提升實驗結果的仿真性。

- **開發適用於虛擬機環境之無線網路測試工具**

  本計畫將利用 QEMU 的 I/O 模擬機制來模擬新的無線網路介面，提供更多元的網路測試環境。

綜合以上所述，本計畫除達成所規劃之各項研究目標與具體的學術與產學合作成果，並完成多項雛型系統之研發。各研究項目將在未來進行更深入的研究與更廣泛的產業合作與推廣，可望能對目前雲端安全技術的發展現況做出貢獻。

陸、 參考文獻

[1] M. Armbrust, A. Fox, R Griffith, A. Joseph, R Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing." in *Communications of the ACM*, Vol. 53, No. 4. April 2010, pp. 50-58.

[2] D. Boneh, G. D. Crescenzo, R. Ostrovsky and G. Persiano, "Public Key Encryption with Keyword Search." in *proceedings of Eurocrypt 2004*, LNCS 3027, 2004, pp. 506-522.

[3] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, "Bigtable: A distributed storage system for structured data." in Proceedings of the *7th USENIX Symposium on Operating Systems Design and Implementation*, 2006.

[4] J. Dean, and S. Ghemawat, "MapReduce: Simplified data processing on large clusters." in Proceedings of *Operating Systems Design and Implementation*, 2004, pp.137-150.

[5] M. J. Freedman, Y. Ishai, B. Pinkas, O. Reingold, "Keyword search and oblivious pseudorandom functions." in *2nd Theory of Cryptography Conference*, volume 3378 of LNCS, 2005, pp 303-324.

[6] S. GHEMAWAT, H. GOBIOFF, and S. -T. LEUNG, "The google file system." in Proceedings of the *nineteenth ACM symposium on Operating systems principles*, 2003, pp. 29–43.

[7] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries." in *5th Theory of Cryptography Conference*, volume 4948 of LNCS, 2008, pp. 155-175.

[8] R. A. Kemmerer, G. Vigna, "Intrusion detection: a brief history and overview." *Computer* Volume: 35 Issue: 4, 2002, pp.27 - 30.

[9] P. Mell and T. Grance, "The NIST Definition of Cloud Computing." in *NIST*, 2009.

[10] J. Oberheide, E. Cooke, and F. Jahanian, "CloudAV: N-Version Antivirus in the Network Cloud." in Proc. of the *17th USENIX Security Symposium*, July 2008.

[11] J. Oberheide, E. Cooke, and F. Jahanian, "Rethinking Antivirus: Executable Analysis in the Network Cloud." in *USENIX Workshop on Hot Topics in Security*, August 2007.

[12] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, "Virtualized In-Cloud Security Services for Mobile Devices." in *Workshop on Virtualization in Mobile Computing*, June 2008.

[13] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks." in *USENIX 13th Systems Administration Conference*, 1999.

[14] D. X. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data." in *IEEE Symposium on Security and Privacy*, 2000.

[15] J. Venner, Pro Hadoop. *Apress*, June 22, 2009.

[16] T. White. 2009. Hadoop: *The Definitive Guide*. O'Reilly Media, Inc. June 2009.

[17] Alexandros G. Dimakis, Brighten Godfrey, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. In Proceedings of the *26th IEEE International Conference on Computer Communications* –INFOCOM 2007, pages 2000–2008. IEEE, 2007.

[18] Y. Wu, A. G. Dimakis, and K. Ranchandran. Deterministic regenerating codes for distributed storage systems. In Proceedings of the *45th annual Allerton conference on Communication, control, and computing*, Allerton'07, pages 1243–1249. IEEE Press, 2007

[19] Alexandros G. Dimakis, Brighten Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 6(9):4539–4551, 2010.

[20] K. V. Rashmi, Nihar B. Shah, P. Vijay Kumar, and Kannan Ramchandran. Explicit construction of optimal exact regenerating codes for distributed storage. In Proceedings of the *47th annual Allerton conference on Communication, control, and computing*, Allerton'09, pages 1243–1249. IEEE Press, 2009.

[21] Nihar B. Shah, K. V. Rashmi, and P. Vijay Kumar. A flexibile class of regenerating codes for distributed storage. In Proceedings of *IEEE symposium on Information Theory* 2010, pages 1943–1947. IEEE Press, 2010.

[22] Soroush Akhlaghi, Abbas Kiani, and Mohammad Reza Ghanavati. A fundamental trade-off between the download cost and repair bandwidth in distributed storage systems. In Proceedings of *IEEE International Symposium on Network Coding* 2010 – NetCod, pages 1–6, 2010.

[23] Salim El Rouayheb and Kannan Ramchandran. Fractional repetition codes for repair in distributed storage systems. In Proceedings of the *48th annual Allerton conference on Communication, control, and computing*, Allerton'10. IEEE Press, 2010.

[24] Alessandro Duminuco and Ernst W. Biersack. Hierarchical codes: A flexible trade-off for erasure codes in peer-to-peer storage systems. *Peer-to-Peer Networking and Applications*, 3(1):52–66, 2010.

[25] Yuchong Hu, Yinlong Xu, Xiaozhao Wang, Cheng Zhan, and Pei Li. Cooperative recovery of distributed storage systems from multiple losses with network coding. Selected Areas in Communications, *IEEE Journal* on, 28(2):268–276, 2010

[26] Frederique Oggier and Anwitaman Datta. Self-repairing homomorphic codes for distributed storage systems. In Proceedings of the 30th IEEE international conference on Computer communications 2011. IEEE Press, 2011.

[27] Theodoros K. Dikaliotis, Alexandros G. Dimakis, and Tracey Ho. Security in distributed storage systems by communicating a logarithmic number of bits. In Proceedings of IEEE symposium on information theory 2010. IEEE Press, 2010.

[28] K. V. Rashmi, Nihar B. Shah, and P. Vijay Jumar. Enabling node repair in any erasure code for distributed storage, 2011.

[29] Sameer Pawar, Salim El Rouayheb, and Kannan Ramchandran. On secure distributed data storage under repair dynamics. Technical Report UCB/EECS-2010-18, University of California Berkeley, EECS, 2010.

[30] Dimitris S. Papailiopoulos and Alexandros G. Dimakis. Distributed storage codes meet multiple-access wiretap channels. In Proceedings of the 48th Annual Allerton Conference on Communication, Control, and Computing, Allerton'10, pages 1420–1427, 2010.

[31] Alexandros G. Dimakis, Vinod Prabhakaran, and Kannan Ramchandran. Decentralized erasure codes for distributed networked storage. IEEE/ACM Transactions on Networking, 14:2809–2816, 2006.

[32] Hsiao-Ying Lin and Wen-Guey Tzeng. A secure decentralized erasure code for distributed network storage. IEEE transactions on Parallel and Distributed Systems, 21:1586–1594, 2010.

[33] Hsiao-Ying Lin and Wen-Guey Tzeng. A secure erasure code based cloud storage system with secure data forwarding. manuscript.

[34] Hsiao-Ying Lin, Wen-Guey Tzeng, and Bao-Shuh Lin. A Decentralized Repair Mechanism for Decentralized Erasure Code based Storage Systems. IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2011.

[35] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at

untrusted stores. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 598–609. CCS ’07 (2007)

[36] Ateniese, G., Di Pietro, R., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: Proceedings of the 4th international conference on Security and privacy in communication networks. pp. 9:1–9:10. SecureComm ’08 (2008)

[37] Ateniese, G., Kamara, S., Katz, J.: Proofs of storage from homomorphic identification protocols. In: Matsui, M. (ed.) Advances in Cryptology ASIACRYPT 2009, Lecture Notes in Computer Science, vol. 5912, pp. 319–333. Springer Berlin / Heidelberg

[38] Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) Advances in Cryptology EUROCRYPT 2005, Lecture Notes in Computer Science, vol. 3494, pp. 562–562. Springer Berlin / Heidelberg

[39] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) Advances in Cryptology CRYPTO 2005, Lecture Notes in Computer Science, vol. 3621, pp. 258–275. Springer Berlin / Heidelberg

[40] Bowers, K.D., Juels, A., Oprea, A.: Hail: a high-availability and integrity layer for cloud storage. In: Proceedings of the 16th ACM conference on Computer and communications security. pp. 187–198. CCS ’09 (2009)

[41] Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: theory and implementation. In: Proceedings of the 2009 ACM workshop on Cloud computing security. pp. 43–54. CCSW ’09 (2009)

[42] Chen, B., Curtmola, R., Ateniese, G., Burns, R.: Remote data checking for network coding-based distributed storage systems. In: Proceedings of the 2010 ACM workshop on Cloud computing security workshop. pp. 31–42. CCSW ’10 (2010)

[43] Curtmola, R., Khan, O., Burns, R.: Robust remote data checking. In: Proceedings of the 4th ACM international workshop on Storage security and survivability. pp. 63–68. StorageSS ’08 (2008)

[44] Curtmola, R., Khan, O., Burns, R., Ateniese, G.: Mr-pdp: Multiple-replica provable data possession. In: Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems. pp. 411–420. ICDCS ’08 (2008)

[45] Damgrd, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) Advances in Cryptology CRYPTO 91, Lecture Notes in Computer Science, vol. 576, pp. 445–456. Springer Berlin / Heidelberg

[46] Erway, C., K¨up ¸c¨u, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. In: Proceedings of the 16th ACM conference on Computer and communications security. pp. 213–222. CCS ’09 (2009)

[47] Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) Advances in Cryptology -EUROCRYPT 2006, Lecture Notes in Computer Science, vol. 4004, pp. 445–464. Springer Berlin / Heidelberg

[48] Juels, A., Kaliski, Jr., B.S.: Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 584–597. CCS ’07 (2007)

[49] Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) Advances in Cryptology -ASIACRYPT 2008, Lecture Notes in Computer Science, vol. 5350, pp. 90–107. Springer Berlin / Heidelberg

[50] Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: Proceedings of the 29th conference on Information communications. pp. 525–533. INFO¬COM’10

(2010)

[51] Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proceedings of the 14th European conference on Research in computer security. pp. 355–370. ESORICS'09 (2009)

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

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

[62] 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.

[63] 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.

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

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

[66] 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.

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

[68] 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.

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

[70] Wikipedia,"Amazon Elastic Compute Cloud," (http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud)

[71] D. Chisnall, The Definitive Guide to the Xen Hypervisor: Prentice Hall PTR, 2007.

[72] G. Reese, Cloud Application Architectures: Building Applications and Infrastructure in the Cloud: O'Reilly, 2009

[73] Virtuatopia,"An Overview of the Hyper-V Architecture," (http://www.virtuatopia.com/index.php/An_Overview_of_the_Hyper-V_Architecture)

[74] J. N. Matthews, E. M. Dow, T. Deshane, W. Hu, J. Bongio, P. F. Wilbur, and B. Johnson, *Running Xen: A Hands-On Guide to the Art of Virtualization*: Prentice Hall, 2008.

[75] R. G. Bace, *Intrusion Detection*: Sams, 2000.

[76] N. Stakhanova, S. Basu, and J. Wong, "A taxonomy of intrusion response systems," *International Journal of Information and Computer Security*, vol. 1, pp. 169-184, 2007.

[77] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *NDSS*, 2003, pp. 253-285.

[78] VMWare,"VMWare Infrastructure Pricing," (http://www.vmware.com/files/pdf/vi_pricing3.pdf)

[79] B. Payne,Georgia Tech, "XenAccess: An Introspection Library for Xen," (http://code.google.com/p/xenaccess/)

[80] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: Malware analysis via hardware virtualization extensions," in *ACM CCS*, 2008, pp. 51-62.

[81] Wikipedia,"Ring (computer security)," (http://en.wikipedia.org/wiki/Ring_%28computer_security%29)

[82] XenSource.com,"XenWindowsGplPv," (http://wiki.xensource.com/xenwiki/XenWindowsGplPv)

[83] T. Henderson and B. Allen,Network World "Xen-based hypervisors push performance limits," (http://www.networkworld.com/reviews/2009/011209-vm-xen-hypervisor-test.html?page=2)

[84] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *SOSP*, 2003, p. 177.

[85] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. Matthews, "Xen and the art of repeated research," in *Usenix*, 2004.

[86] I. Ivanov,"API hooking revealed," (http://www.codeproject.com/kb/system/hooksys.aspx)

[87] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, "Panorama: Capturing system-wide information flow for malware detection and analysis," in *ACM CCS*, 2007, pp. 116-127.

[88] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection through vmm-based out-of-the-box semantic view reconstruction," in *ACM CCS*, 2007, p. 138.

[89] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena, "BitBlaze: A new approach to computer security via binary analysis," *Information Systems Security*, pp. 1-25.

[90] F. Bellard,"QEMU," (http://en.wikipedia.org/wiki/QEMU)

[91] A. Garcia,"Comparing virtualization software performance: QEMU vs UML vs KVM," (http://blogs.igalia.com/berto/2007/06/27/comparing-virtualization-software-performance-qemu-vs-uml-vs-kvm/)

[92] VMWare,"A Performance Comparison of Hypervisor," (http://www.vmware.com/pdf/hypervisor_performance.pdf)

[93] Z. Amsden, D. Arai, D. Hecht, A. Holler, and P. Subrahmanyam, "VMI: An interface for paravirtualization," in *Linux Symposium*, 2006.

[94] AMD,"AMD-V Nested Paging," (http://developer.amd.com/assets/NPT-WP-1%201-final-TM.pdf)

[95] D. Ott,"Virtualization and Performance: Understanding VM Exits," (http://itknowledgehub.com/tag/vm-exit/)

[96] VMWare,"Performance Evaluation of Intel EPT Hardware Assist," (http://www.vmware.com/pdf/Perf_ESX_Intel-EPT-eval.pdf)

[97] B. Cogswell and M. Russinovich, "RootkitRevealer," Available online as www. sysinternals. com/ntw2k/freeware/rootkitreveal. shtml.

[98] S. Sparks and J. Butler, ""Shadow Walker¨: Raising the bar for rootkit detection," in *Black Hat Japan*, 2005, p. 504¡V533.

[99] B. Payne, M. Carbone, and W. Lee, "Secure and flexible monitoring of virtual machines," in *ACSAC*, 2007, pp. 385-397.

[100] S. Jones, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "VMM-based hidden process detection and identification using Lycosid," in *Usenix International Conference On Virtual Execution Environments*, 2008, pp. 91-100.

[101] Tripwire.com,"Tripwire," (http://www.tripwire.com/)

[102] tcpdump.org,"LibPcap Library," (http://www.tcpdump.org/)

[103] Snort,"Snort Network IDS," (http://www.snort.org/)

[104] ClamAV,"ClamAV AntiVirus," (http://www.clamav.net/lang/en/)

[105] TrendMicro,"Trend Micro Accelerates Dynamic Datacenter Security Strategy with Acquisition of Third Brigade," (http://trendmicro.mediaroom.com/index.php?s=43&item=714)

[106] TrendMicro,"Trend Micro Deep Security," (http://us.trendmicro.com/us/solutions/enterprise/security-solutions/virtualization/deep-security/)

[107] T. Benzel et al., "Experience with DETER: A Testbed for Security Research," in Proceedings of Tridentcom. IEEE, 2006, pp. 10–10.

[108] C. Neuman, C. Shah, and K. Lahey, "Running Live Self-Propagating Malware on the DETER Testbed," in Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test, June 2006.

[109] B. White et al., "An Integrated Experimental Environment for Distributed Systems and Networks," in Proc. of the Fifth Symposium on Operating Systems Design and Implementation. Boston, MA: USENIX Association, Dec. 2002, pp. 255–270.

[110] Eide et al., "Integrated Scientific Workflow Management for the Emulab Network Testbed," in Proceedings of the 2006 USENIX Annual Technical Conference, June 2006, pp. 363–368.

[111] R. Bajcsy et al., "Cyber Defense Technology Networking and Evaluation," Commun. ACM, vol. 47, no. 3, pp. 58–61, 2004.

[112] C. Yu and H. Kai, "Collaborative Detection and Filtering of Shrew DDoS Attacks using Spectral Analysis," Journal of Parallel and Distributed Computing, vol. 66, no. 9, pp. 1137–1151, June 2006.

[113] C. Yu, H. Kai, and K. Wei-Shinn, "Collaborative Detection of DDoS Attacks over Multiple Network Domains," IEEE Transactions on Parallel and Distributed Systems, June 2007.

[114] B. White et al., "An Integrated Experimental Environment for Distributed Systems and Networks," ACM SIGOPS Operating Systems Review, vol. 36, pp. 255–270, 2002.

[115] T. Benzel et al., "Design, Deployment, and Use of the DETER Testbed," in DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test 2007. USENIX Association, 2007, pp. 1–1.

[116] K. Sklower and A. D. Joseph, "Very Large Scale Cooperative Experiments in Emulab-Derived Systems," in DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test 2007. USENIX

Association, 2007, pp. 12–12.

[117] J. Mirkovic et al., "Automating DDoS Experimentation," in DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test 2007. USENIX Association, 2007, pp. 4–4.

[118] J. Mirkovic et al., "Measuring Denial of Service," in Proceedings of the 2nd ACM Workshop on Quality of Protection. ACM, 2006, pp. 53–58.

[119] J. Mirkovic et al., "Benchmarks for DDoS Defense Evaluation," in Proceedings of MILCOM. IEEE, 2006, pp. 1–10.

[120] V. Agarwal, "A Scalable Implementation of a Wireless Network Emulator," Master's thesis, University of Utah, 2006.

[121] "Network Simulator 2." [Online]. Available at http://www.isi.edu/nsnam/ns/

[122] "Tribe Flood Network 2000." [Online]. Available at http://ca.com/tw/securityadvisor/virusinfo/virus.aspx?ID=8542

[123] "NetStumbler." [Online]. Available at http://www.netstumbler.com/

[124] "Simple Wireless Scanner." [Online]. Available at http://www.swscanner.org/

[125] T. Ristenpart, E. Tromer, H. Shacham, S. Savagem, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds, " in the Proceedings of the 16th ACM conference on Computer and communications security, pp. 192-212.

[126] Onur Aciicmez , Jean-Pierre Seifert, Cheap Hardware Parallelism Implies Cheap Security, Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography, p.80-91, September 10-10, 2007

[127] Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2/

[128] Amazon Web Services. Auto-scaling Amazon EC2 with Amazon SQS. http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1464.

[129] Amazon Web Services. Creating HIPAA-Compliant Medical Data Applications with Amazon Web Services. White paper, http://awsmedia.s3.amazonaws.com/AWS_HIPAA_Whitepaper_Final.pdf, April 2009.

[130] Amazon Web Services. Customer Agreement. http://aws.amazon.com/agreement/

[131] Paul Barham , Boris Dragovic , Keir Fraser , Steven Hand , Tim Harris , Alex Ho , Rolf Neugebauer , Ian Pratt , Andrew Warfield, Xen and the art of virtualization, Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA

[132] Xen 3.0 Interface Manual. Available at http://wiki.xensource.com/xenwiki/XenDocs.

[133] T. Redkar, "Windows Azure Platform," Expert's Voice in .Net, 2010. (ISBN:1430224797)

[134] Lmbench. Available at http://www.bitmover.com/lmbench/.

[135] Netperf. Available at http://www.netperf.org/netperf/.

[136] G. Judd and P. Steenkiste, "Repeatable and Realistic Wireless Experimentation Through Physical Emulation," SIGCOMM Comput. Commun. Rev., vol. 34, no. 1, pp. 63–68, 2004.

[137] J. Kirch, "Virtual machine security guidelines," The Center for Internet Security, Tech. Rep., 2007.

[138] H.-Y. Tsai, M. Siebenhaar, A. Miede, Y. Huang, and R. Steinmetz, "Threat as a service?: Virtualization's impact on cloud security," IT Professional, vol. 14, no. 1, pp. 32–37, 2012.

國科會補助計畫衍生研發成果推廣資料表

日期:2012/10/30

國科會補助計畫	計畫名稱: 總計畫(2/2)
	計畫主持人: 曾文貴
	計畫編號: 100-2218-E-009-003- 學門領域: 資訊

無研發成果推廣資料

100 年度專題研究計畫研究成果彙整表

計畫主持人：曾文貴	計畫編號：100-2218-E-009-003-
計畫名稱：前瞻性雲端安全儲存、偵測、行為分析與觀測--總計畫(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	100%		
		專書	0	0	100%		
	專利	申請中件數	2	2	100%	件	
		已獲得件數	0	0	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（本國籍）	碩士生	3	3	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	1	1	100%		
國外	論文著作	期刊論文	9	9	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	13	13	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	1	1	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。)	本年度已邀請多位國際知名學者至國內進行技術交流與演講。

	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)

雲端服務是 IT 產業的重點技術，雲端安全更顯得格外重要。本計畫提出之建構方案包含以下四個研究領域：多功能雲端安全儲存、全系統層次動態惡意程式行為分析、即時雲端入侵偵測反制、與雲端網路實驗觀測平台。

在學術成就方面，本計畫之研究成果合計已發表九篇國際期刊論文與十餘篇國際會議論文，足見本計畫在理論研究上具有相當之學術價值。

在技術創新方面，本計畫已建構九個雛型子系統，各具技術優勢，且其中部分技術已與中華電信、友訊科技、趨勢科技、宏達電子(HTC)、喬鼎資訊等業界翹楚簽訂產學合作。部分技術並於專利申請中，足見本計畫之成果在技術上具有創新性與實用性。

在社會影響方面，本計畫已於 0penFoundry 開啟專案並公布部分雛型系統之原始碼，冀望能將安全技術研發與自由軟體社群連結，並有機會引入自由軟體社群的研發能量以吸引產業界投入的意願、厚植國內資訊安全技術的研發潛力。