

# 基於虛擬機器外部觀察與映像檔比對的惡意程式分析

## Malware Behavior Analysis based on Virtual Machine Introspection and Snapshot Comparison

王繼偉<sup>1</sup>、王嘉偉<sup>2</sup>、許家維<sup>3</sup>、謝續平<sup>4</sup>

交通大學資訊科學與工程學系

E-mail : [cwwangabc@gmail.com](mailto:cwwangabc@gmail.com)<sup>1</sup>, [glacier314@gmail.com](mailto:glacier314@gmail.com)<sup>2</sup>,  
[vicnobody@gmail.com](mailto:vicnobody@gmail.com)<sup>3</sup>, [ssp@csie.nctu.edu.tw](mailto:ssp@csie.nctu.edu.tw)<sup>4</sup>

### 摘要

目前新型的惡意程式，大多會試圖以各種技巧如 Rootkit、Hooking 等，躲避偵測程式的分析，造成分析人員追蹤的困難。面對這些問題，專業的資安人員往往缺乏系統化、自動化的平台工具，快速的進行分析程序並作出修補或更正。本論文透過一模擬的 X86 主機系統，比對檢測目標對系統某些重要區域，如檔案系統、登錄機碼、驅動程式、程序以及執行緒等，所造成的前後差異。且透過由外部直接觀察虛擬機器內部的 Virtual Machine Introspection 技術，得知某些資訊是否為內部的惡意程式所變造，以偵測惡意程式的 Rootkit 行為。

**關鍵詞：**惡意程式、虛擬機器、Rootkit、軟體安全。

### 一、介紹

未知惡意程式的分析，一直是資安領域中深具影響力研究項目。對於目前每天變化出的無數病毒、蠕蟲以及木馬等威脅，在目前防毒軟體皆無法提供準確偵測的情況下，資安人員往往必須檢測某檔案的行為，以判斷目標是否具有危害資安的危險行為。但如今新型的惡意程式，大多會使用許多的混淆或隱藏手法，以躲避偵測程式的分析。甚至亦有惡意程式藉由破壞或干擾的方式，令偵測軟體的功能失效，而提高分析工作的困難度。此外，在檢測目標行為的過程中，分析者常常必須藉由多個不同的工具程式，才能側錄到該檔案全面性的行為，由於這種過程需要高度的人工參與，也降低了分析工作的效率。

本研究試圖透過運行檢測目標於虛擬機器內部的的方式，再透過觀察虛擬機器的各項資訊來分析惡意程式的行為。首先，當檢測目標在虛擬機器內部開始運行後，必然會對系統狀態造成改變，例如檔案系統的存取、登錄機碼的修改、啟動某應用程

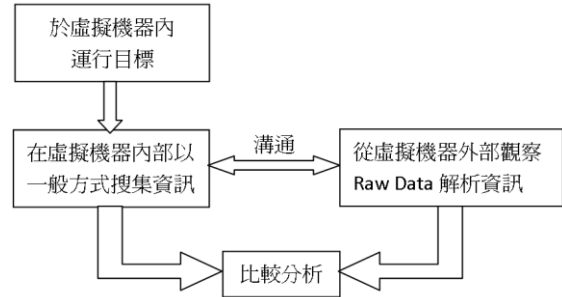


圖 1 使用內外比對來分析惡意程式

式，甚至於註冊新的驅動程式或寫入開機磁區等等。因此透過比對虛擬機器前後狀態的不同，我們可以很快地得知，檢測目標對系統造成的修改部分。這種觀察方式的一大好處是可以得到全面性的完整資訊，而不必以各種不同的工具輔助分析，這是本研究所試圖使用的第一種分析手法。

而我們所提出的第二種分析方法為，將從外部觀察虛擬機器 (Virtual Machine Introspection, VMI) 得到的資訊，與由虛擬機器內部的普通 User 進行觀察所得的結果進行比對，找出不一致的部分。這些不一致的部分所代表的意義為，系統內部是否有隱藏或混淆行為的存在，如 Rootkit 或 Hooking 等。這些行為的主要目的，在於造成系統內部使用者對某資訊的認知，與真實的系統狀況造成差異。舉例來說，一個企圖隱藏某運行中程序的 Rootkit，勢必會在使用者企圖列舉系統中所有程序時，修改甚至破壞列舉的清單結果，然而該程序是的確存在的，只要透過適當的 VMI 技術，便能從虛擬機器外部得知。因此，透過如圖 1 這種內外比對的手法，便能確切得知 Rootkit 行為的存在與否。而目前日益猖獗的 Rootkit 手法，反而能提供我們偵測惡意軟體上的一大切入點。這些被隱藏的資訊越多，這種觀測方式所能得到的資訊也就越多。

本研究的具體貢獻為，提出由虛擬機器內外部分別觀察的方式，以得知 Rootkit 行為的存在，以及開發一自動化的惡意程式行為分析平台，以加速分析工作速度。在第二章中我們將介紹與本論文相關的現有研究成果，第三章則敘述開發系統的設計理念與架構，我們亦對本系統的分析準確性以真實的惡意軟體樣本加以評估，分析討論敘述於第四

This work was supported in part by TRUST Center of UC Berkeley, ITRI, Institute for Information Industry, Chung Shan Institute of Science and Technology, Chungwa Telecomm., Investigation Bureau, Dlink, the International Collaboration for Advancing Security Technology (iCAST) and Taiwan Information Security Center (TWISC), respectively.

章。文章最後我們將對本研究作一結論。在附錄中則附上所開發系統對病毒樣本的分析結果。

## 二、相關研究

使用虛擬機器來分析惡意程式的有效性極高，因此這種手法成為分析人員常用的技術之一[2][3][4]，以上研究並未著重於 Rootkit 行為的偵測。而 VMI 的概念則由[1]所提出，在我們的研究中所使用到的 VMI 的功能，由許多概念與其類似。而我們所使用的虛擬機器，則是透過修改 QEMU[5]已達成我們要的目的。

## 三、系統設計與架構

Rootkit 的根本目的在於隱藏自身，企圖讓使用者無法發覺其存在進而延長其存活壽命。因此無論 Rootkit 採取什麼樣的技術，其根本目的在於造成系統內部使用者對某資訊的認知，與真實的系統狀況造成差異。此外，儘管正常的偵測程式會修改某些重要的系統部件，但這種隱藏行為並不常見於這種偵測程式。因此我們認為這種行為十分適合作為鑑識時的判斷依據。

要找出這種「由系統內部觀點取得的資訊，與真實系統狀況不一致」的行為，最重要的一點為如何從系統外部去了解真實的系統狀況。為了達成這一點，我們認為透過將系統虛擬化，以虛擬機器的觀點是最有希望的解決手法。透過在虛擬機器內執行目標程式有以下幾點好處：首先，目標程式與真實機器有較好的隔離性，不至於在分析時令惡意程式入侵至真實系統，降低分析準確性甚至對系統造成破壞。其次，由於虛擬機器對外部的真實系統來說只是單一的程序，其 CPU、記憶體與硬碟狀態都十分容易取得並進行分析。

由於使用虛擬機器來分析惡意程式的有效性極高，因此這種手法成為分析人員常用的技術，也因此有惡意程式開始會對其所在的執行環境進行檢查，確定並不是在虛擬環境下後才執行自身的惡意行為。對於這一點我們分析如下：首先，使用虛擬技術來分析惡意程式乃當今最重要的技術之一，儘管有部份惡意程式開始進行虛擬環境的偵測，鑑識人員也不應立即放棄此種有效的分析手法。其次惡意程式在進行虛擬環境的偵測時，往往只做非常簡單的特徵比對，由於虛擬軟體的多樣性，這種作法讓以往讓偵測軟體失效的「特徵碼不足」問題，從偵測軟體開發人員移至惡意程式作者身上，提高惡意程式的寫作難度。最後，由於我們可以完全掌握虛擬出的環境，因此要改變虛擬環境的特徵以躲避惡意程式的偵測相對較容易。使用以上的觀點進行研究，若能發現某項資訊的不一致，則可判定為某種匿蹤行為。

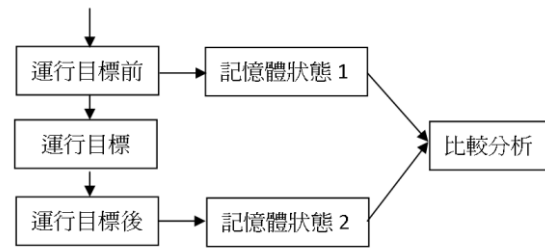


圖 2 使用前後比對來分析惡意程式

要從系統內部搜集某項資訊的方式較為直接，只需要模仿一般的使用者層級應用程式來向系統發出查詢即可。此處查詢的方式應儘可能貼近越上層的應用程式，因為 Rootkit 的最主要目的即為攔截這些查詢並假造變更查詢結果以矇騙使用者，我們希望搜集到的恰好就是這些假的資訊，以期與我們從系統外部解析到的資訊有所不同。上述方法可找出 Rootkit 的隱藏行為，但仍有許多其他的 Rootkit 行為值得我們注意，例如 Rootkit 不一定會以程序的方式執行，有時 Rootkit 可能修改啟動磁區 MBR，或是以驅動程式方式掛載至系統內部。這些行為的共同模式為對系統內部的資料結構進行修改，以在 Windows 作業系統中掛載入核心的驅動程式為例，都必須要向系統註冊對應的 `_DRIVER_OBJECT` 以及 `_DRIVER_EXTENSION` 資料結構，因此若能找出目標執行後記憶體中多出的這些資料結構，便有助於 Rootkit 行為的判定。

要能判定這些行為，我們認為較準確的方式為，分析比較目標執行前與執行後的系統記憶體狀態，以找出前後的差異。此處對系統狀態的判讀，若從系統內部進行，則有可能受到 Rootkit 行為的干擾導致找不出變動的地方，因此一樣必須從系統外部進行解析。整體流程如圖 2。

其中一困難點在於如何從虛擬機器外部解析出真實的資訊，即所謂的 Virtual Machine Introspection。從機器外部看到的資訊，缺乏系統內部對資料進行解析的抽象意義，例如從外部觀察該虛擬機器的磁碟映像檔，所看到不是檔案與目錄結構，而是一塊塊磁區的資料。同理，由外部觀察虛擬機器的記憶體狀態，也無法看到程序的虛擬位址空間，而是一大塊的物理層記憶體。

要從記憶體中取出有用的資訊，其中一項有力的技術為，辨認記憶體中的資料結構。在 Windows 核心中，各種資料皆以 Kernel Object 的方式存在，並在不同的函式或部件中傳遞，以執行緒為例，要讓 Windows 在進行 Context Switch 時分配時間給某一執行緒，則該執行緒必須要有一 `_ETHREAD` 的資料結構在等候 Context Switch 的連結串列中，而這個資料結構中的許多欄位，其值可能永遠為常數或固定範圍，因此我們可以透過這特徵辨識出這些資料結構，以得到最精確的狀態資訊。在截取資

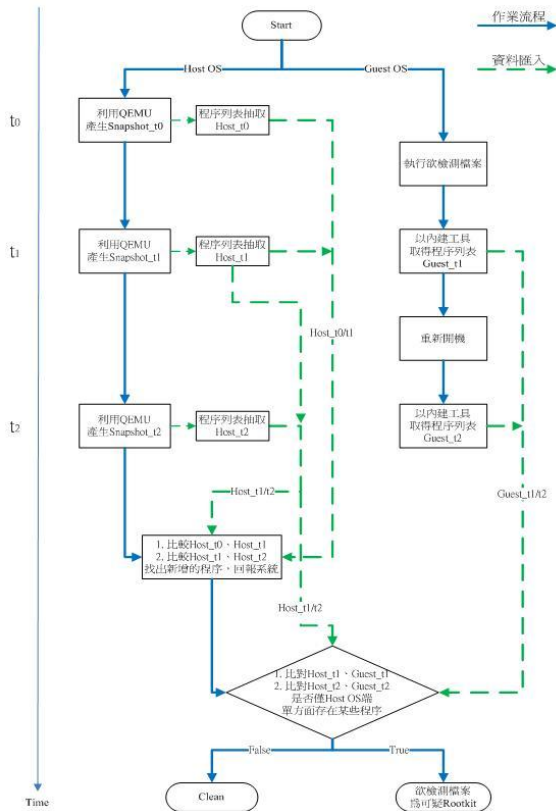


圖 3 新增程序與隱藏程序掃描

料結構的部份，我們將參考 Volatility[7]這套 Open Source 的記憶體鑑識工具的作法，來幫助我們找出有意義的資訊。以下我們依 Rootkit 行為分別介紹，要偵測該行為所需進行的分析工作。

### 程序隱藏

為了達到偵測所有程序，進一步判斷是否遭受 Rootkit 感染，此實做系統將針對兩個方向進行分析：是否有新增的未知程序、是否有隱藏程序存在，並基於虛擬機器的應用上，根據虛擬機器內所得到的程序資訊以及從虛擬機器外取得系統資料並分析後的資訊，進一步比對兩方程序紀錄是否存在差異，藉此來偵測出是否有可疑未知程序產生以及隱藏程序的存在。我們將在 Linux 環境下以 QEMU 建立虛擬機器，當欲檢測之檔案輸入時，將其運行在虛擬機器上，再利用 QEMU 的 Snapshot 功能來協助實做此系統，系統架構簡圖如圖 3：

Guest OS 剛運行時，是為時間點  $t_0$ ，Host OS 先對 Guest OS 做 Snapshot 紀錄記憶體資料，接著在 Guest OS 上執行我們所要檢測的檔案後，經過一參數時間  $t$  後於時間點  $t_1(=t_0+t)$  Host OS 重複上述的紀錄動作，同時 Guest OS 也利用自身系統 API 取得程序列表。之後我們將 Guest OS 重新開機，待系統穩定後於時間點  $t_2$ ，Host、Guest OS 再重複時

間點  $t_1$  之紀錄動作。

擁有運行檔案前後時間點的紀錄後，我們將針對前述兩個主要方向進行分析。在偵測是否有未知程序新增的部分，Host OS 端先呼叫本檢測系統內的函式將 Snapshot 所得的 raw memory data 進行分析，並萃取出記憶體內所有程序資訊整理成列表，再以  $t_0$ 、 $t_1$  以及  $t_1$ 、 $t_2$  兩階段時間點所得的程序列表紀錄兩兩進行比對，過濾出每個階段系統內新產生了哪些程序，並回報給本檢測系統提供進一步的判斷。

偵測 Rootkit 的部分，Host OS 以上述萃取出來的程序列表，與 Guest 端之紀錄兩兩比對  $t_1$ 、 $t_2$  時間點的紀錄，觀察是否有差異存在。若欲檢測之檔案內含 Rootkit 惡意程式，則執行後 Client OS 所使用的程序監測工具已經遭到修改或置換，因此並無法取得真正的程序列表，然而所有的程式運行時都必須先載入至記憶體方可執行，因此 Host OS 透過 QEMU 的 Snapshot 可以取得正在虛擬機器上運作的 Guest OS 其完整的記憶體資料，加以分析、萃取出程序資訊後，便可供我們觀察所有正在運行的所有程序，包含 Rootkit 隱藏程序。因此當我們將 Host、Guest OS 所紀錄的資料相互比對後，雙方在  $t_1$  以及  $t_2$  時間點的紀錄檔理應要相同，若只有 Host OS 單方面透過 QEMU 紀錄萃取出來的資料可觀察到部份程序的存在，此時我們便認定欲檢測的檔案確實內含 Rootkit 程式碼企圖隱藏自己，並可能進行惡意的行為。

### 檔案修改及隱藏

由於使用者無法從現有的工作管理員，清楚地知道這些檔案被開啟，所以我們預期掃描出所有正被開啟的檔案名稱，並以前後比對的方列出差異點。如同之前所敘，為了得到記憶體的資訊，我們同樣利用虛擬機器來實做。首先，一樣得到兩個不同時間點的記憶體資訊，利用記憶體分析的方式，找出已開啟的檔案名稱，並且找出對應的程序識別(process ID)。對時間先後順序做比較，列出在執行目標程式後所開啟的檔案名稱。雖然有可能會列出跟目標程式不同的執行程序所開啟的檔案，不過我們仍保留這些資訊，以便往後的分析。

為了找出系統內是否有被 rootkit 隱藏的惡意檔案或是主要開機磁區是否有被修改遭到植入 rootkit，我們使用 TSK (The Sleuth Kit) 來協助我們偵測 rootkit 的行為。此 Library 可以觀察虛擬硬像檔的檔案系統裡所有資訊，包含有目前硬碟裡的 metadata、有那些檔案，檔案的內容等等。為了找出被隱藏的檔案：我們

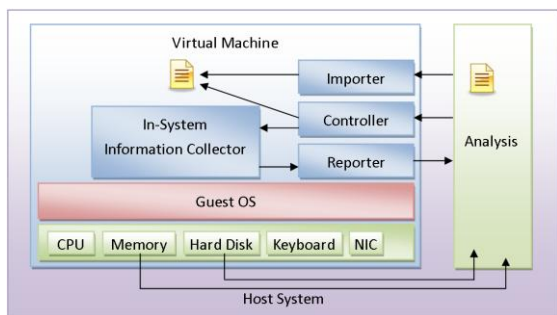


圖 4 惡意程式分析平台架構

透過修改 QEMU 於其執行過程中，記錄曾變動過的磁區編號，再利用 TSK 查詢硬碟映像檔的檔案配置表反推出擁有該磁區的檔案名稱及檔案路徑，再去 Guest OS 利用一般的 Win32 API 查詢是否有該檔案的存在。若沒有，則表示該檔案被隱藏。在找出開機磁區是否有被修改的部分則相對簡單，我們只需觀察那些被修改過的磁區編號中是否有開機磁區的編號即可。

#### 登錄機碼隱藏

如前述，Rootkit 往往會 Hook 某些用來查詢登錄檔的 Win32 API 或系統呼叫，以避免某些機碼發現。因此，要觀察登錄檔最準確的方式，就是完全不靠這些函式的輔助，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。

我們分析的步驟如下，首先我們將儲存登錄資訊的幾個重要 Hive 檔案，從虛擬機器的硬碟映像檔中抽取出來。以上的抽取動作皆由系統外部進行，而不經由系統內部的 API 或系統呼叫。當以上檔案從硬碟映像檔中抽取出來後，我們將解析這些 Hive 檔案，並列出某些重要登錄目錄下的所有機碼值，之後由系統內部的觀察程式，以 Win32 API 去取得同樣目錄下的所有機碼值，並與系統外部的觀察結果進行比對，以確認是否有隱藏機碼的 Rootkit 行為。

#### 新增驅動程式

由於目前 Rootkit 程式皆以驅動程式的方式掛載進系統核心，以取得系統最高的執行權限，因此在我們的分析項目中，也會對虛擬機器的核心層級記憶體部分進行掃描，找出是否有新增的驅動程式項目。

為了達成以上的分析構想，並且自動化整體的分析流程，我們設計一惡意程式行為分析平台，以 QEMU 為主體架構進行修改，加上外部的 VMI 組件，並在 Guest OS 中加入與外部溝通或自動化用的軟體。整體系統架構如圖 4。以觀察並在分整體的系統架構

本系統將使用虛擬機器模擬出一 X86 的主機系統，並預先準備一安裝有 Windows XP 作業系統環境的磁碟映像檔 S0，我們將保持 S0 的零污染性，亦即確保 S0 並未被任何惡意程式所感染，同時在不失一般性的狀況下，儘量減少其上運行的應用程式數目，以避免背景干擾。而我們的分析程式主體，則佈署在虛擬機器外部，亦即與該虛擬機器程式處於同一階層，以進行從外部觀察虛擬機器狀態的工作。同時，在虛擬機器內部，我們將安裝一個由我們自己撰寫用以搜集內部資訊的 In-System Information Collector、放入待檢測目標的 Importer、接收從外部而來的命令加以執行的 Controller，以及傳回結果給外部的 Reporter。

由於部分惡意軟體會妨礙正常程式的行為如網路資料傳輸，避免包含自身存在事實的系統資訊曝光，藉此來保護自身的安全。因此我們在虛擬機器內外之間建立一塊固定且共享的記憶體區塊，當 Collector 收集資訊完畢便將結果寫入該共享區塊，最後僅發送一硬體訊號給虛擬硬體以通知外部前往讀取。因惡意軟體無法得知該記憶體區塊為內外共享且發送的硬體訊號未包含任何內部資訊，Collector 所收集的結果將在惡意軟體未察覺的安全管道下傳輸。

系統運作整體流程如下，分析人員指定待檢測檔案後，系統將以 S0 的狀態開始運行，並透過 Importer 將待檢測檔案置入虛擬機器內。之後 Controller 將運行該檢測檔案，在執行該檔案五分鐘後，進行一次 snapshot 得到系統狀態 S1。我們的分析工作將基於 S0、S1 這兩個系統狀態，以及透過 Reporter 截取到的資訊進行。

#### 四、實驗評估

為了測試我們所設計的系統，在面對實際的惡意程式與 Rootkits 時是否能提供有效的分析結果，我們以數個實際網路上流傳的惡意程式，進行分析測試。樣本包含：隨身碟病毒 KAVO、WOW PW Stealer、Stone bootkit[8](攻擊開機磁區)以及 FUTO rootkit[6]等。礙於篇幅，我們僅將隨身碟病毒 KAVO 與 Stone bootkit 的詳細分析結果列於附錄。以下僅敘述最明顯的惡意程式行為分析結果。

在病毒 KAVO 部分，我們側錄到以下項目：

==== Registry Diff Scanning ====
HKCU/Software/Microsoft/Windows/CurrentVersion/Run/kava C:\WINDOWS\system32\kavo.exe
==== Disk Diff Scanning ====
/WINDOWS/system32/kavo.exe /WINDOWS/system32/kavo0.dll /Documents and Settings/dsns/Local Settings/Temp/wdagnb7.dll
==== Driver Diff Scanning ====
0x00f8928000 0x006000 wincab.sys

在上面的分析結果中，我們可觀察到檢測目標在系統登錄檔的開機啟動項目中，新增了一個機碼，以達成開機自動執行的目的。同時，在檔案系統中亦新增了幾個可疑的檔案。在附錄中的檔案系統變更項目所列出的，除了新增的檔案外，還包含所有被修改過的檔案。此外我們也掃描出了系統核心中新掛載的驅動程式。而這些行為也與該 KAVO 病毒的行為相符合。

在 PW Stealer 部分，我們側錄到以下項目：

==== Registry Diff Scanning =====
<b>HKLMACHINE\SOFTWARE\Microsoft\Windows/CurrentVersion/Run/Torjan Program C:\WINDOWS\WINLOGON.EXE</b>
==== Disk Diff Scanning =====
/WINDOWS/finder.com /WINDOWS/system32/finder.com /WINDOWS/1.com /WINDOWS/ExERoute.exe /WINDOWS/system32/MSCONFIG.COM /WINDOWS/system32/regedit.com /WINDOWS/Debug/DebugProgram.exe /WINDOWS/system32/command.pif /WINDOWS/explorer.com /WINDOWS/system32/rundll32.com /WINDOWS/system32/dxdiag.com /Program Files/Internet Explorer/iexplore.com /WINDOWS\WINLOGON.EXE /Program Files/Common Files/iexplore.pif

從上面的分析結果可得知 WOW PW Stealer 也會新增開機啟動項目，並將自身拷貝出多份副本到檔案系統中。

在 Stone bootkit 部分，我們側錄到以下項目。此樣本的檢測結果顯示我們的分析平台可檢測出對開機磁區的修改行為。

==== Disk Diff Scanning =====
/Stoned/Drivers/Sinowal Extractor.sys /Stoned/Master Boot Record.bak /Stoned/Applications/Sinowal Loader.sys /Stoned/Applications/Hibernation File Attack.sys /Stoned/Applications/Forensic Lockdown Software.sys /Stoned/Applications/Windows.sys //\$Secure:\$SDH //\$Secure:\$SDS /Stoned/Drivers/Sinowal.sys /Stoned/Drivers/Black Hat Europe 2007 Vipin Kumar POC.sys
==== MBR Modification Scanning =====
<b>MBR Modified</b>

在 FUTO rootkit 部分，必須要加以說明的是，由於 FUTO 本身並不會一隻惡意程式，它的功能是隱藏別的記憶體中的程式，因此我們的實驗設計中是以 FUTO 來隱藏 notepad.exe 這隻已經運行的程序。在 FUTO 執行後我們側錄到以下項目：

==== Driver Diff Scanning =====
<b>0x00f7a40000 0x010000 msdirectx.sys</b>
==== Process Diff Scanning =====
<b>notepad.exe</b>

從以上的分析結果可充分顯示我們的分析平台對惡意程式的登錄修改、檔案系統修改、驅動程式註冊、以及 Rootkit 隱藏程序行為皆可進行完整的自動分析側錄。

## 五、結論

針對目前對資安造成嚴重威脅的惡意程式，目前分析時往往必須結合多種工具，以 Ad-hoc 的方式進行繁雜的掃描，以得到全面性的行為側錄資訊。此外，以往於系統內部進行偵測分析的方式，往往會受到惡意程式的干擾或破壞失去效用。有別人過往的分析法，本研究所提出的虛擬機器快照比對與外部觀測技術，可迅速準確分析出檢測目標的登錄修改、檔案系統修改、驅動程式註冊、以及 Rootkit 隱藏程序行為，且避免被惡意程式干擾運作的可能性。而基於這些技術所建構的惡意程式行為分析平台，將可在資安人員進行軟安檢測時，作為一有力的分析工具，加速分析項目的進行。

## 參考文獻

- [1] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," *In Proceedings of the Network and Distributed Systems Security Symposium*, February 2003.
- [2] A. Goel, K. Po, K. Farhadi, Z. Li, and E. de Lara, "The Taser Intrusion Recovery System," *In Proceedings of the twentieth ACM Symposium on Operating Systems Principles*, pages 163–176, New York, NY, USA, 2005. ACM.
- [3] Carsten Willems, Thorsten Holz, and Felix Freiling, "Toward Automated Dynamic Malware Analysis Using CWSandbox," *IEEE Security and Privacy*, vol. 5, no. 2, pp. 32-39.
- [4] U. Bayer, A. Moser, C. Kruegel, and E. Kirda, "Dynamic Analysis of Malicious Code," *Journal in Computer Virology*, 2006.
- [5] F. Bellard, "QEMU, a Fast and Portable Dynamic Translator," *Proceedings of the 2005 USENIX Annual Technical Conference*, April 10–15, 2005, Anaheim, CA, USA
- [6] FUTO.  
[http://www.openrce.org/articles/full\\_view/19](http://www.openrce.org/articles/full_view/19)
- [7] Volatility  
<https://www.volatilesystems.com/default/volatility>
- [8] N. Kumar and V. Kumar, "Vboot Kit: Compromising Windows Vista Security," *In Black Hat Europe*, Amsterdam, March 2007.

## 附錄

### 分析案例：隨身碟病毒 KAVO

```
===== Registry Diff Scanning =====
BOTH:+ >
HKEY_CURRENT_USER/Software/Microsoft/Windows/CurrentVersion/Run/kava
C:\WINDOWS\system32\kavo.exe^M
===== Disk Diff Scanning =====
/WINDOWS/system32/config/SysEvent.Evt
/Documents and Settings/dsns/NTUSER.DAT
/Documents and Settings/dsns/Local Settings/History/History.IE5/index.dat
//SMFT
/Documents and Settings/dsns/NTUSER.DAT.LOG
/WINDOWS/system32/config/system
/Documents and Settings/dsns/Cookies/index.dat
* /Documents and Settings/dsns/Local Settings/Temporary Internet Files/Content.IE5/O5Y7GL2Z/97ED7CEAE39F51B123B73CC29A684[1].jpg
/WINDOWS/system32/config/system.LOG
/WINDOWS/system32/config/software
/WINDOWS/system32/kavo.exe
/System Volume
Information/_restore{BF386772-CD88-4BB7-A880-E86FCA469157}/RP2/change.log
/WINDOWS/system32/kavo0.dll
/WINDOWS/system32/config/software.LOG
/Documents and Settings/dsns/Local Settings/Temporary Internet Files/Content.IE5/index.dat
//LogFile
* /Documents and Settings/dsns/Local Settings/Temporary Internet Files/Content.IE5/O5Y7GL2Z/black_oo[1].gif
/WINDOWS/Temp/Perflib_Perfdata_794.dat
/Documents and Settings/dsns/Local Settings/Temp/wdagnb7.dll
//Bitmap
===== Driver Diff Scanning =====
\??\C:\WINDOWS\system32\wincab.sys
0x00f8928000 0x006000 wincab.sys
===== Process Diff Scanning =====
===== MBR Modification Scanning =====
```

### 分析案例：Stone bootkit

```
===== Registry Diff Scanning =====
===== Disk Diff Scanning =====
/Stoned/Drivers/Sinowal Extractor.sys
/Stoned/Master Boot Record.bak
/Stoned/Applications/Sinowal Loader.sys
/Documents and Settings/dsns/NTUSER.DAT
/Stoned/Applications/Hibernation File Attack.sys
//SMFT
/Stoned/Applications/Forensic Lockdown Software.sys
/Documents and Settings/dsns/NTUSER.DAT.LOG
//$Secure:$SII
/Stoned/Applications/Windows.sys
* /Documents and Settings/dsns/Local Settings/Temporary Internet Files/Content.IE5/UFU34XOP/mail[2].&am=!Sg304q2GSLi5Bbvy0fc6Qqw-V8VA6EHLdxRDhrc92XJp
* /Documents and Settings/dsns/Local Settings/Temporary Internet Files/Content.IE5/$5AFCLQ3/E32B13BB5DEF4B39A0A04E4EA92F6D[1].gif
* /Documents and Settings/dsns/Local Settings/Temporary Internet Files/Content.IE5/O5Y7GL2Z/97ED7CEAE39F51B123B73CC29A684[1].jpg
/WINDOWS/system32/config/software
/WINDOWS/system32/config/software.LOG
/System Volume
Information/_restore{BF386772-CD88-4BB7-A880-E86FCA469157}/RP2/change.log
//$Secure:$SDH
//$Secure:$SDS
//LogFile
/
/Stoned/Drivers/Sinowal.sys
/Stoned/Drivers/Black Hat Europe 2007 Vipin Kumar POC.sys
//Bitmap
===== Driver Diff Scanning =====
===== Socket Diff Scanning =====
===== Process Diff Scanning =====
===== MBR Modification Scanning =====
MBR Modified
```