# 基於 Tripwire 檢測工具以偵測變形 Rootkit 之研究

**林明孝**
**大葉大學資訊管理系**
**E-mail: crab_109@yahoo.com.tw**

## 摘要

現今的網際網路發展迅速且成熟，駭客的人數不斷的快速成長，而入侵行為也漸趨變化與複雜，駭客得以針對Linux或Windows作業系統的漏洞與弱點來發展出多樣化的攻擊技術，如：Rootkit惡意軟體。

因此，本研究從 Linux 系統管理者的角度進行探討，找出作業系統是否已被植入 User Mode Rootkit。在所設計的偵測機制中，首先使用 Chkrootkit 工具偵測目前已知的 Rootkit，再針對此已知 Rootkit 入侵特性藉由 Tripwire 來檢測系統檔案的完整性，以從資料庫中找出被變形 Rootkit 所異動的項目，再與已知型 Rootkit 所會異動的項目做比對，因而找出變形 Rootkit，藉此提昇變形 Rootkit 惡意軟體之偵測能力。本論文最後亦實際模擬測試，以驗證所提偵測機制之高效益。

**關鍵字：**入侵偵測系統、Rootkit、Linux

## 1. 前言

對於成功攻擊系統的入侵者而言，為了不被系統管理者發現，想辦法隱藏自己的蹤跡是相當重要的工作。系統一旦被入侵並取得管理者權限時，系統上的記錄檔是否還是正確，往往是一個很大的疑問。若要補足現有主機型入侵偵測的弱點，得從作業系統本身來討論，於是本研究使用 Linux 系統的記錄檔經由 Tripwire [11],[13]檢測工具來偵測其檔案的完整性，藉以判斷是否為非授權的使用者竄改。所以要根本的解決檔案被異動的問題，設計其偵測機制是必要且刻不容緩的。

然而，最早出現的 Rootkit 主要是針對 Linux 作業系統，目前的 Rootkit 可分為兩種類型：User Mode 與 Kernel Mode，前者能讓駭客再次的入侵系統並竄改系統程式，其所竄改的系統程式會隱藏駭客的程序、檔案與目錄等等，讓系統管理者難以發現其蹤跡，而後者 Kernel mode Rootkit 會竄改系統核心與記憶體，當核心被竄改時也無法全部復原，即使復原則系統也不被管理者所信任。基於上述，本研究針對 User Mode Rootkit 進行探討，並分析已知 Rootkit 的類型、特性與攻擊行為，以有效的偵測工具來偵測已知與變形 Rootkit 且檢測系統檔案的完整性，進而達到偵測變形 Rootkit，藉此提昇變形 Rootkit 惡意軟體之偵測能力，並利用電子郵件通知系統管理者，以便系統管理者能快速的將遭受到攻擊的檔案復原。

## 2. 文獻探討

以下將針對 Rootkit 探討、相似度的計算、記錄檔特徵選取與現有的偵測技術作一說明：

### 2.1 Rootkit 探討

以下針對 Rootkit 的原理與應用作一說明：

(1) Rootkit 的原理是由四個攻擊步驟所構成 [6]，其攻擊步驟說明如下：

步驟 1：弱點分析：Rootkit 會搜尋作業系統、提供服務的伺服端程式(Server Process)或網路服務等等是否有安全上的漏洞，可以使用弱點掃描來檢視一下，當測試的伺服端程式有回應時，就代表有安全上的漏洞。

步驟 2：利用弱點：攻擊者會利用系統的弱點而設計出程式碼(code) ，程式碼所提供的服務為跳板或產生一個執行環境。

步驟 3：隱藏蹤跡：當利用弱點入侵系統時，會先取得 root 權限並從系統記錄檔中消除所有

攻擊的軌跡，可以讓攻擊者再進行攻擊的過程中，隱藏攻擊者的檔案及目錄，之後便是隱藏自己的蹤跡並保留下一次使用root帳號的權限。

步驟4：建立後門：當攻擊者取得系統存取權限時，即使原來的弱點尚在未被移除，還是會安裝一個或多各後門程式(backdoor)。

(2) 再針對User mode與Kernel mode兩種Rootkit的應用作一說明：

- User mode Rootkit是最常見到的Rootkit，也是最廣泛被拿來使用的工具，其使用方法是會替換或隱藏於系統中正常的應用程式與系統檔案，其應用為隱藏攻擊者程式、後門程式、木馬程式與監聽程式 [1],[7]。
- Kernel mode Rootkit [5]為Kernel層級中最強而有力的Rootkit，比User mode Rootkit更有力，利用LKM(Loadable Kernel Module)的功能可讓攻擊者來攻擊系統，已經成為最難偵測的 Rootkit，其應用為隱藏程序、隱藏網路連結、隱藏LKM的信號、利用LKM傳遞訊息與改變檔案的執行。

## 2.2 相似度的計算

入侵偵測的樣本特徵通常都是非量化特徵，所以使用傳統的空間距離公式來計算其樣本相似度是會影響結果，因欲將質化特徵進行量化動作時，依靠的都是研究者的主觀定義，而這些主觀定義的特徵值，通常都會帶有些許的誤差。

Oh及Kim在2004年提出一種不同於以往空間距離的概念來計算樣本相似度 [8]。其算式如公式(1)所示，其中$S_i, S_j$代表兩連續資料樣本，$|E_i \cap E_j|$代表兩樣本中任一對項目(item)交集的個數，而$|E_1|$及$|E_2|$代表該樣本中任一對項目的總數，舉例來說：若$S_1 = \{A,B,C,D\}$、$S_2 = \{A,C,D,E\}$，則$E_1 = \{AB, AC, AD, BC, BD, CD\}$，$|E_1| = 6$，$E_2 = \{AC, AD, AE, CD, CE, DE\}$，$|E_2| = 6$，$E_1 \cap E_2 = \{AC, AD, CD\}$，$|E_1 \cap E_2| = 3$，而$S_1$與$S_2$的相似度$Sim(S_1, S_2)$為1/2。

$$Sim(S_i, S_j) = \frac{|E_i \cap E_j|}{\frac{|E_i| + |E_j|}{2}} \quad (1)$$

## 2.3 記錄檔特徵選取

根據Forrest [14]等學者利用system call來產生某一服務正常行為應有的特徵值，但在這過程中值得考慮許多該服務運行的因素，例如：不同資料輸入，會有不同system call的產生，且費時又無法完全保證能收集到全部正常的特徵值。而這些收集到的正常特徵值也只適用該版本的服務，不同版本、不同服務還得從新來過，較沒效率。

從使用者歷史指令記錄檔中，可以去了解該使用者的常態行為，故可適用於偵測本機端攻擊，但對於遠端攻擊偵測較沒有幫助，況且有些服務是不須使用者登入到本機端就可以存取。

另外，雖然比對檔案的雜湊值可以了解那些檔案有被異動，卻無法從中得知問題的來源，因而難以解決系統上的漏洞。但是經由比對檔案的雜湊值之後，系統中那些記錄檔有被非授權的使用者異動過，可以藉此了解是否有入侵行為，再判斷是否已經被駭客入侵。

因此，為了較容易尋找問題的來源以及可以提供駭客入侵的證據，本研究欲就系統記錄檔與檔案的雜湊(hash)值來進行分析，並使用Oh及Kim的相似度計算方式，有了正確的樣本特徵及相似度計算方法，再搭配上合適的記錄檔特徵比對演算法，就可以得到較佳的偵測結果。

## 2.4 現有偵測技術

目前偵測rootkit的技術可區分為五大類 [12]：交叉察看偵測、硬體式偵測、行為偵測、特徵偵測及完整性偵測。就以上述五種偵測技術的優缺點來作一說明：

1. 交叉察看偵測(Cross view based detection)
   - ■ 優點：可檢測系統檔案(File)、程序(Process)與機碼(Registry key)。
   - ■ 缺點：只應用於Windows系統的偵測技術
2. 硬體式偵測(Hardware detection)

■ 優點：

(1)擁有自己的CPU，並透過DMA對記憶體掃描。

(2)具有一定程度的偵測成果。

■ 缺點：需花費大量的成本建置與維護。

3. 行為偵測(Behavioral detection)

■ 優點：

(1)能夠辨識已知或未知的病毒或惡意軟體。

(2)不需要更新其特徵或病毒碼到資料庫中。

■ 缺點：

(1)所設定的行為標準不易界定，容易造成誤報(false positive)。

(2)無法識別特定的病毒或惡意軟體。

4. 特徵偵測(Signature based detection)

■ 優點：

(1)快速且準確辨別現有已知的病毒或惡意軟體。

(2)不會發生誤報(false positive)或假警報(false negative)。

(3)只需新增新的病毒碼或特徵到資料庫中，即可偵測出新的病毒或惡意軟體。

■ 缺點：

(1)須大量的資料庫。

(2)無法偵測未知型的病毒或惡意軟體。

5. 完整性偵測(Integrity based detection)

■ 優點：

(1)可快速的發現系統檔案是否被感染。

(2)不需更新特徵或病毒碼到資料庫中。

(3)可有效的偵測系統是否被非授權使用者入侵。

■ 缺點：

(1)無法辨認惡意病毒或惡意軟體行為。

(2)若系統的檔案異動頻繁，則容易引起誤報。

綜合上述的分析比較，基於為了能有效偵測與降低儲存成本，是故本研究將使用特徵偵測與完整性偵測來設計偵測機制，以使有效偵測變形 Rootkit。

## 3. 基於 Tripwire 檢測工具以偵測變形 Rootkit

本研究所設計的基於 Tripwire 檢測工具以偵測變形 Rootkit 機制，主要由偵測流程、記錄檔特徵比對演算法的設計來著手。

### 3.1 偵測流程之設計

本論文在 Linux 系統上結合 Chkrootkit 與 Tripwire 兩個偵測工具，並設計出偵測變形 Rootkit 的偵測機制，針對這兩個偵測機制的回報檔(Response)來產生警訊，並通知系統管理者能夠做後續的處理。以下介紹本研究所提出之基於 Tripwire 檢測工具以偵測變形 Rootkit 之偵測機制，其偵測流程如圖 1 所示：

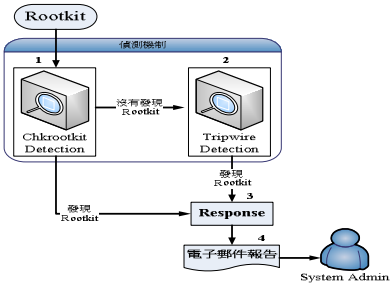


圖 1 系統偵測流程

經由 Chkrootkit Detection 檢測系統且並未偵測到 Rootkit，再進一步使用本論文所提出之偵測機制中的 Tripwire Detection，可藉以偵測變形的 Rootkit，如圖 2 所示；其相關步驟如下所示：

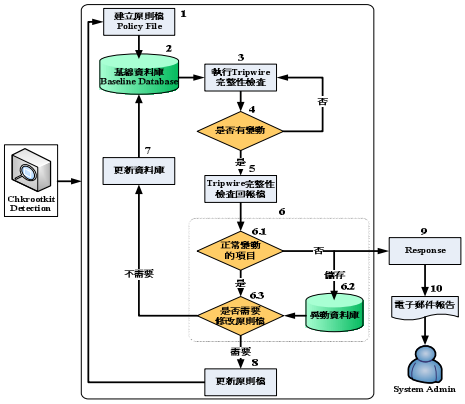


圖 2 基於 Tripwire 之偵測流程

步驟 1：在 Tripwire Detection 建置時，需要先建立原則檔(Policy File)，並設定要對系統檔案做哪些的檢查動作，將此規則寫到原則檔內。

步驟 2：當原則檔建立完之後，依據原則檔的內容來建立基線資料庫(baseline database)，其會依據原則檔規則將檔案的資訊儲存下

來，提供往後比對之用。基線資料庫只需建立一次即可，不需要每次使用 Tripwire 時都建立基線資料庫，只需要將其內容更新即可。

步驟 3：執行 Tripwire 完整性檢查(Integrity Check)，會依據基線資料庫所儲存的內容與欲檢查的對象做比對，判斷是否有檔案被異動。

步驟 4：根據檢查結果判斷檔案是否有被異動過。如果有發現異動，表示已遭到未經授權者竄改系統檔案則執行步驟 5；反之，如果需再重新確認系統檔案則執行步驟 3。

步驟 5：Tripwire 完整性回報檔會把 Tripwire 檢測的結果，以回報檔(Report File)來報告。

步驟 6：就針對 Tripwire 完整性檢查回報檔內的資訊來做「正常的變動項目」、「儲存異動資料庫」與「是否需要修改原則檔」的判斷項目逐一說明。

步驟 6.1：經由 Tripwire 完整性檢查回報檔來判斷系統檔案是否為正常的異動，如果是的話表示是經由正當權限所進行的動作，則必須進行步驟 6.3；反之，則同時進行步驟 6.2 與步驟 9。

步驟 6.2：經由 Tripwire 完整性檢查回報檔的判斷為非正常的異動項目，並將其儲存於另外建立的異動資料庫中。系統管理者可從異動資料庫中找出被變形 Rootkit 所異動的項目，再與現有 Rootkit 的異動項目比對，藉此判斷異動項目中是否有所關聯，在 3.2 節中將針對資料庫的比對作一詳述。

步驟 6.3：在重複的進行完整性檢查與資料庫更新的過程中，有時也會增加一些新的檢查對象、檔案或變更檢查項目(property)。在更新的資料中如需修正原則檔，則執行步驟 8；反之，只需更新特徵資料庫則執行步驟 7。

步驟 7：當有新的檢查對象、檔案或變更檢查項目，則將其更新到特徵資料庫中。

步驟 8：當有新的檢查對象、檔案或變更檢查項目，則將其更新到原則檔中。

步驟 9：經由 Tripwire 在系統中所偵測到的非正常異動，進而產生的警訊會記錄到回報檔(Response)。

步驟 10：將回報檔(Response)的訊息利用電子郵件傳送給系統管理者(System Admin)，能夠提供系統管理者做後續的處理。

### 3.2 記錄檔特徵比對演算法

本研究使用Oh及Kim的 [8]相似度計算方法(Oh & Kim, 2004)，經由Tripwire完整性檢查回報檔的判斷為非正常的異動項目，並將其儲存於另外建立的異動資料庫中。系統管理者可從異動資料庫中找出被變形Rootkit所異動的項目，再與現有Rootkit的異動項目比對，藉此判斷異動項目中是否有所關聯，以下針對資料庫的比對作一詳述:

輸入：主要為門檻值(Sim_Threshold)ST、已知Rootkit資料庫K_D、變形(Metamorphic)Rootkit資料庫M_D。

輸出：則為相似的特徵群並將其加入法則庫中。

步驟1：依據資料庫K_D與資料庫M_D中的樣本來計算其相似度，其作法如下:

$$Sim\ (S_i, S_j) = \frac{\frac{|E_i \cap E_j|}{|E_i| + |E_j|}}{2}$$

其中$S_i$, $S_j$分別代表變形與已知Rootkit，$|E_i \cap E_j|$則代表兩Rootkit特徵中任一對項目(item)所交集的個數，而$|E_i|$及$|E_j|$各代表變形與已知Rootkit特徵中任一項目的總數。

步驟2：將樣本先取得群集編號$S_i$。

步驟3：依據相似度比對：

$$Sim\ (S_i, S_j) > \frac{ST}{2}$$

，檢視兩個特徵是否高於門檻值。

步驟4：將$S_j$的特徵相似度高於門檻值ST，則設為同一群C；反之，因較低的相似度會影響其結果，所以不予考慮，則設為Outlier。

步驟 5：依其群集 C 內各 $S_j$ 的特徵來與 $S_i$ 作判別，若有不同的特徵則依圖 2 之步驟 6.3 更新原

則檔(policy)或更新資料庫，以供 Tripwire 作後續檢測之用；反之，則為已知特徵，並不需更新原則檔(policy)或更新資料庫。來判別是否會產生原則，若原則產生將存入原則庫，否則回到步驟 3。

因此，為了較容易尋找問題的來源以及可以提供駭客入侵的證據，本研究欲就系統記錄檔與檔案的雜湊(hash)值來進行分析，並使用 Oh 及 Kim 的相似度計算方式，有了正確的樣本特徵及相似度計算方法，再搭配上合適的記錄檔特徵比對演算法，就可以得到較佳的偵測結果。

## 4. 系統實作與實驗分析

我們以實驗證明所提之基於 Tripwire 檢測工具以偵測變形 Rootkit 機制是可提高偵測之效益。實驗攻擊以已知 Rootkit，如：ark、Balaur、Dica、Fuckit、t0rn 與變形 Rootkit，如：cb-rootkit、toolkit、bashdoor 來進行測試，再分別以現有的三項偵測工具，如：Chkrootkit [1]、Rkhunter [9]、rootcheck [10]與本機制進行測試比較。以 cb-rootkit Rootkit 為例，分別使用三項偵測工具與本機制進行測試比較，說明如下：

cb-rootkit：

以 cb-rootkit 植入 Linux 系統，其可入侵系統並取得系統資訊，最後完成植入後門，如圖 3 所示。以偵測工具與本機制來掃描系統，從中會發現系統有些檔案已經被 Rootkit 所感染，只有本機制可偵測到此 Rootkit 植入系統中，如圖 7 所示，而 Chkrootkit、Rkhunter、Rootcheck 雖可偵測到系統遭受到感染，但卻偵測到 SHV5、Showtee 此 Rootkit 進而造成誤判，如圖 4 (粗線)、圖 5(粗線)、圖 6(粗線)所示。


圖 3 以 cb-rootkit 植入系統畫面

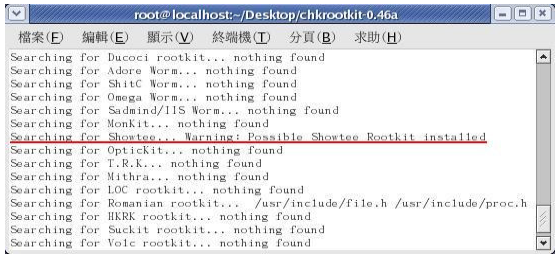


圖 4 以 Chkrootkit 偵測 cb-rootkit 畫面

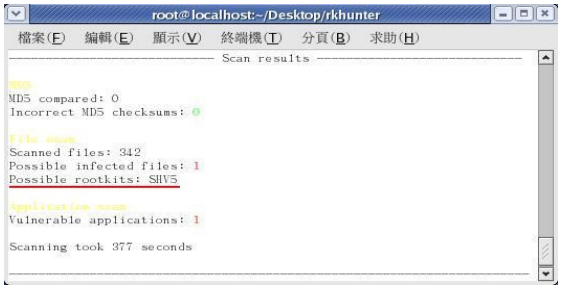


圖 5 以 Rkhunter 偵測 cb-rootkit 畫面

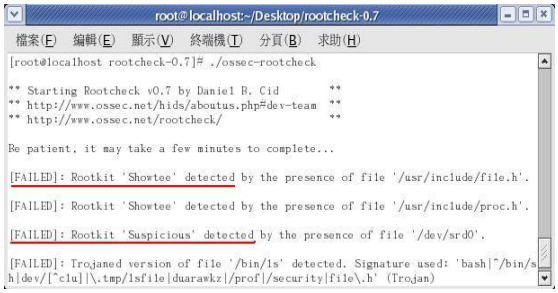


圖 6 以 Rootcheck 偵測 cb-rootkit 畫面

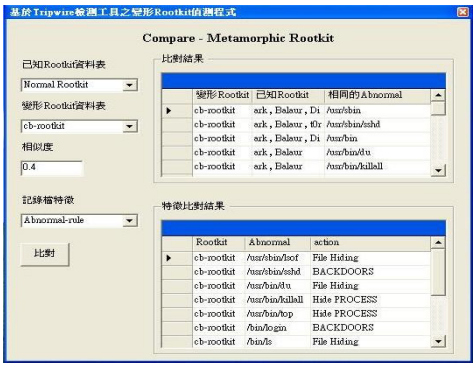


圖 7 以本機制偵測 cb-rootkit 畫面

經由上述 Rootkit 的偵測結果，發現現有的偵測機制並不能完全的偵測到現有或變形的 Rootkit，因其偵測機制是以現有 Rootkit 所感染的特徵來比對，雖然能在系統中找出被 Rootkit 所感染的記錄檔，但是卻無法正確的判別此 Rootkit，容易造成誤判，由此可以了解到本論文所提出的基於 Tripwire 檢測工具以偵測變形 Rootkit 機制，的確有較高的偵測結果，如表 1 所示。

表 1 與其他偵測機制比較

| 偵測工具 / Rootkit | Chkrootkit | Rkhunter | Rootcheck | 本機制 |
|---|---|---|---|---|
| ark | O | O | X | O |
| Balaur | △ | O | △ | O |
| Dica | △ | O | △ | O |
| Fuckit | △ | O | X | O |
| t0rn | O | △ | O | O |
| cb-rootkit | △ | △ | △ | O |
| toolkit | X | O | X | O |
| bashdoor | X | X | X | O |

O 代表可偵測到　X 代表未能偵測到　△代表誤判

現有的三種偵測工具皆能偵測出系統遭受到 Rootkit 感染，但是卻不能完全的判別其 Rootkit，而造成偵測工具的誤判，而本論文提出之「記錄檔特徵比對演算法」可藉由 Rootkit 的異動項目與記錄檔的特徵來比對，經由實驗結果得知其他三項現有的偵測工具都不如本論文所提出之「記錄檔特徵比對演算法」。

在優越性比較中，本論文比較了 Chkrootkit、Rkhunter、Rootcheck 三項偵測工具，在 Rootkits 的偵測結果上，本論文皆有著一定的優勢，也確認本論文提出的「基於 Tripwire 檢測工具以偵測變形 Rootkit 機制」，的確有相當的優越性。

## 5. 結論

本研究設計一基於 Tripwire 檢測工具以偵測變形 Rootkit 機制，這個偵測機制的作用為可偵測現有已知的 Rootkit 且能偵測變形的 Rootkit，藉由多種的偵測技術能夠更加確保系統安全的運作，且能有效提升系統管理者之判斷能力與 Linux 系統之檔案完整性，其得到一些研究貢獻，如下所示：

1. Rootkit 的類型、特性與攻擊行為之分析：

   經由研究後可得知現有 Rootkit 的類型(User mode 與 Kernel mode)的攻擊特性與行為以發現其攻擊原理，可提供管理者更深入了解 Rootkit。

2. 獲致一套適合 Linux 系統所使用的 Rootkit 偵測機制：

   本研究使用 Linux 系統的記錄檔當來源資料，並依所研究的 Rootkit 類型來設計其偵測機制，整合為一適合 Linux 系統所使用的 Rootkit 偵測機制。

3. 藉由其偵測機制之系統記錄檔的完整性來判斷是否有入侵行為，並可達到偵測變形的 Rootkit：

   使用 Tripwire 來檢測系統記錄檔的完整性，藉由其判斷是否有被植入 Rootkit 且被未授權的使用者竄改，再根據異動資料庫的資料來判斷是否是為變形的 Rootkit 異動。

4. 使用本機制可讓系統管理者能夠快速的將遭受到攻擊的 Linux 系統檔案復原：

   基於上列三項成果的整合與搭配，得到此基於 Tripwire 檢測工具以偵測變形 Rootkit 之偵測機制，已達到偵測變形 Rootkit 的功能，讓系統管理者能快速的將遭受到攻擊的系統檔案復原。

## 參考文獻


[1] B. Andreas, “UNIX and Linux based Rootkits Techniques and Countermeasures”, https://www.dfn-cert.de/team/bunten/rootkits_first 2004.pdf

[2] Chkrootkit, http://www.chkrootkit.org

[3] S. Jha and M. Hassan, “Building Agents for rule-based intrusion detection system,” *Computer Communications*, Vol. 25, No. 15, pp. 1366-1373, 2002

[4] S. T. King and P. M. Chen, “Backtracking Intrusions,” *ACM Transactions on Computer Systems(TOCS)*, Vol. 23, No. 1, pp. 51-76, 2005.

[5] C. Kruegel, W. Robertson and G. Vigna, “Detection Kernel-Level Rootkits Through Binary Analysis,” *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC)*, 2004.

[6] W. E. Kuhnhauser, “Root kits: An operating systems viewpoint,” *ACM SIGOPS Operating Systems Review*, Vol. 38, No. 1, pp. 12-23, 2004.

[7] J. Levine, B. Culver and H. Owen, “A Methodology for Detecting New Binary Rootkit Exploits,” *Proceedings IEEE SouthEastCon 2003*, 2003.

[8] S.J Oh and J.Y. Kim, “A Hierarchical Clustering Algorithm for Categorical Sequence Data,” *Information Processing Letters*, Vol. 91, No. 3, pp. 135-140, 2004.

[9] Rkhunter, http://www.rootkit.nl/.

[10] Rootcheck, http://www.ossec.net/en/rootcheck.html

[11] R. F. DeMara and A. J. Rocke, “Mitigation of network tampering using dynamic dispatch of mobile agents,” *Computers & Security*, vol. 23, no. 1, pp. 31 – 42, 2004.

[12] Security Focus : http://www.securityfocus.com/infocus/1854

[13] Tripwire, http://www.tripwire.com.

[14] A. Somayaji, and S. Forrest, Automated Response Using System-Call Delays”. *Proceedings of the 9th Usenix Security Symposium*, pp. 185-197.