201508534

(19)中華民國智慧財產局

(12)發明說明書公開本 (11)公開編號：TW 201508534 A

(43)公開日： 中華民國 104 (2015) 年 03 月 01 日

(21)申請案號：102130367 (22)申請日： 中華民國 102 (2013) 年 08 月 23 日

(51)Int. Cl.： *G06F21/55 (2013.01)* *G06F21/56 (2013.01)*

(71)申請人：國立交通大學 (中華民國) NATIONAL CHIAO TUNG UNIVERSITY (TW)
新竹市大學路 1001 號

(72)發明人：沈宗賢 SHEN, ZONG SHIAN (TW) ；謝續平 SHIEH, SHIUH PYNG (TW)

(74)代理人：蔡清福

申請實體審查：有 申請專利範圍項數：10 項 圖式數：9 共 43 頁

(54)名稱

產生純化惡意程式的方法、偵測惡意程式之方法及其系統

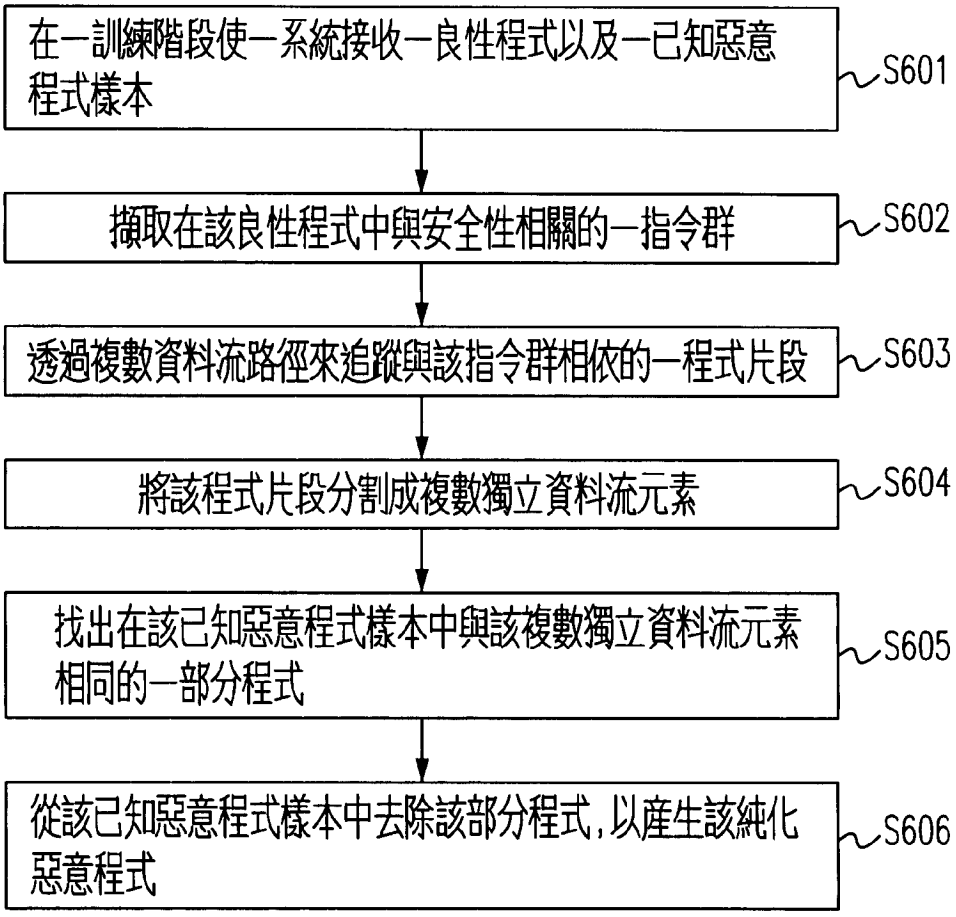
METHOD OF GENERATING DISTILLATION MALWARE PROGRAM, METHOD OF DETECTING MALWARE PROGRAM AND SYSTEM THEREOF

(57)摘要

本發明提出一種在一系統中產生一純化惡意程式的方法，該方法包含下列步驟：在一訓練階段使該系統接收一良性程式以及一已知惡意程式樣本。擷取在該良性程式中與安全性相關的一指令群。透過複數資料流路徑來追蹤與該指令群相依的一程式片段。將該程式片段分割成複數獨立資料流元素。找出在該已知惡意程式樣本中與該複數獨立資料流元素相同的一部分程式。從該已知惡意程式樣本中去除該部分程式，以產生該純化惡意程式。

The invention provides a method of generating a distillation malware program in a system. The method comprises steps of: causing the system to receive a benign program and a known malware program sample in a training procedure; extracting an instruction set associated with security in the benign program, tracing a program slice associated with the instruction set by a plurality of data flow path; slicing the program slice into a plurality of independent data flow elements; indentifying a same part program between the known malware program sample and the plurality of independent data flow elements; and removing the same part program from the known malware program sample for generating the distillation malware program.

TW 201508534 A



第六圖

201508534

發明摘要

※ 申請案號：102130367

※ 申請日：102.8.23 ※IPC 分類：G06F 21/55 (2013.01)
G06F 21/56 (2013.01)

【發明名稱】(中文/英文)

產生純化惡意程式的方法、偵測惡意程式之方法及其系統/METHOD OF GENERATING DISTILLATION MALWARE PROGRAM, METHOD OF DETECTING MALWARE PROGRAM AND SYSTEM THEREOF

【中文】

本發明提出一種在一系統中產生一純化惡意程式的方法，該方法包含下列步驟：在一訓練階段使該系統接收一良性程式以及一已知惡意程式樣本。擷取在該良性程式中與安全性相關的一指令群。透過複數資料流路徑來追蹤與該指令群相依的一程式片段。將該程式片段分割成複數獨立資料流元素。找出在該已知惡意程式樣本中與該複數獨立資料流元素相同的一部分程式。從該已知惡意程式樣本中去除該部分程式，以產生該純化惡意程式。

【英文】

The invention provides a method of generating a distillation malware program in a system. The method comprises steps of: causing the system to receive a benign program and a known malware program sample in a training procedure; extracting an instruction set associated with security in the benign program, tracing a program slice associated with the instruction set by a plurality of data flow path; slicing the program slice into a plurality of independent data flow elements; indentifying a same part program between the

known malware program sample and the plurality of independent data flow elements; and removing the same part program from the known malware program sample for generating the distillation malware program.

【代表圖】

【本案指定代表圖】：第（六）圖。

【本代表圖之符號簡單說明】：

無

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

發明專利說明書

（本說明書格式、順序，請勿任意更動）

【發明名稱】（中文/英文）

產生純化惡意程式的方法、偵測惡意程式之方法及其系統/METHOD OF GENERATING DISTILLATION MALWARE PROGRAM, METHOD OF DETECTING MALWARE PROGRAM AND SYSTEM THEREOF

【技術領域】

【0001】 本發明是關於一種惡意程式之偵測方法及其系統，特別是關於一種基於靜態分析的惡意程式之偵測方法及其系統。

【先前技術】

【0002】 近年來行動裝置的使用上已超越個人電腦。在個人電腦上對於系統的惡意程式之攻擊同樣地也出現在行動裝置上，例如手機或平板電腦上對於安卓作業系統的惡意程式之攻擊。惡意程式的橫行使得行動裝置的使用者曝露在惡意程式的攻擊之威脅下，而惡意程式的快速演變亦導致防毒軟體公司無法全面地協助使用者防範惡意攻擊。因此不論產業界或學術界皆需投入更多的資源以研發可有效偵測惡意程式之技術。

【0003】 惡意程式偵測技術是近年來資安重要的一環。由於惡意程式迅速演變並發窄出各式混淆技術，因此精確的偵測方法有其必要性。在個人電腦上的系統所發展出來的惡意程式偵測之技術者主要可分爲靜態分析與動態分析兩種方法。於靜態分析的方法中，藉由擷取待測程式之控制流程而模塑其結構性特徵是常用的技術；然而，頻繁出現於惡意程式之良性程式特徵容易混淆此類技術。於動態分析的方法中，監控以及模塑待測程式之執行期行爲亦是經常使用的技術；然而此類技術受限於行動裝置模擬器之擬真技術，無法有效觀察惡意程式與電信服務之相關行爲，亦無法觀

察惡意程式與行動裝置特有元件之間的相關行爲。

【0004】 在具有公開號20120222120的美國發明專利申請中，揭露了藉由監控相關應用程式介面(Application Program Interface, API)所觸發的活動來識別惡意行爲。在此揭露中，在桌上型個人電腦惡意程式的偵測可應用到行動裝置上；然而，在行動裝置上的惡意程式的指令群普遍地與系統共用的指令群被共同包裝起來。因此藉由整個程式的流程或是一小片段的程式之流程來分析惡意程式是否具備遂行惡意之行爲將面臨到準確率不高的問題，若是未將系統共用的指令群所構成之程式片段從惡意程式中去除則容易造成誤報的情形。

【0005】 在具有公開號20120072988的美國發明專利申請中，揭露了在收集惡意程式而產生控制流程與資料流的模型，並將此模型應用於偵測未知的程式。此揭露還提供一種超級區塊(super block)的新技術用以更穩健地產生流程圖模型，但是此發明並未解決惡意程式中與系統共用的指令群所構成程式流程片段對真正惡意程式造成干擾之問題。

【0006】 在具有公開號20100011441的美國發明專利申請中，揭露了在減輕惡意程式中被植入的混淆之人爲程式碼，以加強偵測的準確率。此揭露著重於解開被加密的惡意程式，並將被解密後的惡意程式轉換成標準的格式。然而，此發明並未解決惡意程式中與系統共用的指令群所構成程式流程片段對真正惡意程式造成干擾之問題。

【0007】 在具有公告號8370931的美國發明專利中，揭露了使用動態行爲吻合的技術來偵測惡意程式。此揭露設計了一種演算法，該演算法藉由比對由一特定的程序造成的系統事件與一組規則來決定是否有任何可疑的行爲。

【0008】 在具有公告號8266698的美國發明專利中，揭露了使用動態

行爲吻合的技術來偵測惡意程式。此揭露收集在客戶端的行爲特徵，以及設計了一種演算法來決定是否運作的應用程式爲惡意程式。

【0009】 前述的先前技術中的惡意程式語意模型都含有其他好的、良性的程式，因此偵測不夠精準。有鑑於此，期待有一種方法可將已知惡意程式樣本模型純化，並採取更爲精準的比對演算法。

【發明內容】

【0010】 依據上述構想，本發明提出一種在一系統中產生一純化惡意程式的方法，該方法包含下列步驟：在一訓練階段使該系統接收一良性程式以及一已知惡意程式樣本。擷取在該良性程式中與安全性相關的一指令群。透過複數資料流路徑來追蹤與該指令群相依的一程式片段。將該程式片段分割成複數獨立資料流元素。找出在該已知惡意程式樣本中與該複數獨立資料流元素相同的一部分程式。從該已知惡意程式樣本中去除該部分程式，以產生該純化惡意程式。

【0011】 依據上述構想，本發明提出一種判斷一待測程式是否包含一惡意程式的系統，該系統接收一良性程式與一已知惡意程式樣本，該系統包含一語意模型擷取子系統、一惡意程式模型純化子系統、以及一惡意特徵比對子系統。該語意模型擷取子系統擷取該良性程式中與安全性相關的一指令群、透過複數資料流路徑來追蹤與該指令群相依的一程式片段、以及將該程式片段分割成複數獨立資料流元素。該惡意程式模型純化子系統將與該複數獨立資料流元素相同的部分從該已知惡意程式樣本中去除，以形成一純化惡意程式。該惡意特徵比對子系統比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式包含該惡意程式。

【0012】 依據上述構想，本發明提出一種判斷一待測程式是否包含遂

行惡意行爲的方法，包含下列步驟：在一非惡意程式中擷取與安全性相關的指令群來建立一語意模型。去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式。比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式具備惡意行爲。

【0013】 依據上述構想，本發明提出一種判斷一待測程式是否包含遂行惡意行爲之指令的方法，包含下列步驟：在一非惡意程式中擷取與安全性相關的第一指令群來建立一語意模型。在一已知惡意程式樣本中擷取與安全性相關的第二指令群，並去除該第二指令群中與該語意模型相同的部分而形成一純化惡意程式。比對該純化惡意程式與該待測程式中與安全性相關的指令群，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行爲之指令。

【0014】 依據上述購想，本發明提出一種判斷一待測程式是否包含遂行惡意行爲之指令的方法，包含下列步驟：爲一非惡意程式建立一語意模型。去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式。比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行爲之指令。

【0015】 藉由本發明之系統與方法，偵測惡意程式的錯誤率下降、偵測率提高、且待測程式與惡意程式的比對更有效率

【圖式簡單說明】

【0016】

第一圖：本發明判斷一待測程式是否包含一惡意程式的系統。

第二圖：本發明透過複數資料流路徑來追蹤與該指令群相依的一程式片段的示意圖。

第三圖：本發明將已知惡意程式樣本純化的示意圖。

第四圖：本發明語意模型擷取子系統的示意圖。

第五圖：本發明待測程式與純化惡意程式107相似度比較的示意圖。

第六圖：本發明在一系統中產生一純化惡意程式的方法的示意圖。

第七圖：本發明判斷一待測程式是否包含遂行惡意行爲的方法的示意圖。

第八圖：本發明判斷一待測程式106是否包含遂行惡意行爲之指令的方法的示意圖。

第九圖：本發明另一較佳實施例判斷一待測程式106是否包含遂行惡意行爲之指令的方法的示意圖。

【實施方式】

【0017】 請參閱第一圖，其爲本發明判斷一待測程式106是否包含一惡意程式的系統10。請參閱第二圖，其爲本發明透過複數資料流路徑來追蹤與該指令群相依的一程式片段的示意圖。請參閱第三圖，其爲本發明將已知惡意程式樣本純化的示意圖。請同時參閱第一、第二、以及第三圖，該系統10接收一良性程式101與一已知惡意程式樣本102，該系統10包含一語意模型擷取子系統103、一惡意程式模型純化子系統104、以及一惡意特徵比對子系統105。該語意模型擷取子系統103擷取該良性程式101中與安全性相關的一指令群1010、透過複數資料流路徑PA, PB, PC來追蹤與該指令群1010相依的一程式片段111、以及將該程式片段111分割成複數獨立資料流元素G1, G2, G3。該惡意程式模型純化子系統104將與該複數獨立資料流元素G1, G2, G3相同的部分從該已知惡意程式樣本102中去除，以形成一純化惡意程式107。該惡意特徵比對子系統105比對該純化惡意程式107與該待測程式106，當兩者相似度達到一特定標準時，則該惡意特徵比對子系統105判斷該待測程式106包含該惡意程式。在另一較佳實施例中，在該惡意特徵比

對子系統105判斷該待測程式106包含該惡意程式之後，該惡意程式模型純化子系統104可將該惡意程式再進一步純化，並將該惡意程式提供至該模型資料庫108，以作爲新的樣本；或是不經過純化直接提供給模型資料庫108以作爲新樣本。

【0018】 在第一圖中，訓練資料集109包含該良性程式101與該已知惡意程式樣本102，良性程式101由該模型資料庫提供108。在一訓練階段時，該良性程式101經由語意模型擷取子系統103來擷取其語意模型，該已知惡意程式樣本101亦經由語意模型擷取子系統103來擷取其語意模型。該待測資料集110包含該待測程式106，在該訓練階段後，該惡意特徵比對子系統104比對待測程式106與該純化惡意程式107它們內部的資料流元素的相似度以判定待測程式106是否有惡意。首先，必須對該良性程式101的位元組碼(byte code)進行反組譯以找出該指令群1010，其中該指令群1010包含一隱私相關的指令群、一未受Dalvik虛擬機器保護的指令群、及一背景自動執行的指令群。對於已知惡意程式樣本102也是同樣進行反組譯以找出另一指令群，其有可能與指令群不同，而本發明主要的目的是剔除該已知惡意程式樣本102中與該良性程式101相同的該指令群1010，以純化該已知惡意程式樣本102。依據各種不同的系統平台，相同的位元組碼經過反組譯後可能會得到不同的指令；一般而言，反組譯後可得組合語言程式碼或機械語言程式碼，接下來便是找出與安全性相關的指令群。

【0019】 應用程式在使用資源時必須獲得存取資源的許可，而該隱私相關的指令群會存取相關於電信服務的資訊或是個人私密資料。因此像是手機的Android作業系統或是IOS作業系統在安裝應用程式時都會詢問存取個人隱私的許可後才安裝應用程式，然而這些對於隱私資料的存取都會影響到資訊安全，故這些隱私相關的指令群的監控管理是重要的。在本發明

中該未受Dalvik虛擬機器保護的指令群以F_P來表示。

【0020】 Dalvik虛擬機器(DVM)是Android作業系統的核心組成部分之一。應用程式由受到DVM保護的指令群來存取作業系統的內部資源時，使用作業系統的裝置或系統將可受到保護，例如，保護其不會被資料塞爆緩衝區(buffer overflow)。該未受Dalvik虛擬機器保護的指令群包含java runtime指令、java native interface (JNI)指令、以及java Class Loader相關指令。java runtime指令可直接呼叫Android作業系統，JNI指令可呼叫原生Linux作業系統，java Class Loader相關指令不僅可呼叫原生Linux作業系統來存取DVM所保護的資源，而且還可存取不屬於DVM所保護的外部資源。在本發明中該未受Dalvik虛擬機器保護的指令群以F_E來表示。

【0021】 關於背景自動執行的指令群，此類指令群可探測裝置的周圍環境，或是自動於背景服務執行，例如背景開啓相機以傳送影像的指令、背景執行側錄按鍵之輸入以竊取密碼的指令等。在本發明中該背景自動執行的指令群以F_R來表示。

【0022】 請參閱第四圖，其爲本發明語意模型擷取子系統103的示意圖。該語意模型擷取子系統103更包含一關鍵點定位器(Critical Point Locator) 1031、一程式切片器(Program Slicer) 1032、以及一資料流元素產生器(Foot Print Generator) 1033。該關鍵點定位器1031主要的功能在於擷取與安全性相關的該指令群1010，其稱爲關鍵點集(Critical Point Set)，該關鍵點集爲安全性相關的語意之由來，並可供進一步分析。該關鍵點集爲該指令集1010，其以I_P來表示，$I_P = I_C \cup I_R$，其中I_C代表$F_P \cup F_E$之指令群，I_R代表一組屬於F_R的指令群。

【0023】 請同時參閱第二圖與第四圖，程式切片器1032的主要功能係使用收集到的關鍵點集來擷取與資料相依的指令群1010，這些資料如同第

二圖中的複數資料變數1, 2, 3, 4, 5, 6, 7，指令群1010如同第二圖中的指令1, 2, 3, 4, 5, 6, 7。在第二圖中總共有複數個資料流路徑，其分別爲資料流路徑PA, PB, PC。這些資料流路徑之追蹤(tracing)可分成兩種，一種是往程式進行步驟的前步驟追溯，稱爲往前追溯切片(Backward Slicing)，另一種是往程式進行步驟的後步驟來找尋，稱爲往後尋找切片(Forward Slicing)，在第二圖中，程式進行的步驟包含經過節點120, 121, 122, 123, 124, 125, 126。程式切片器1032係針對該複數資料流路徑PA, PB, PC中的每一個資料變數往前追溯其來源資料，並找出執行該來源資料的所有路徑；或針對該複數資料流路徑PA, PB, PC中的每一個資料變數往後查找其後續執行的資料，並找出執行該後續執行的資料的所有路徑。例如，第二圖中的複數資料變數5係由指令5來執行，複數資料變數5的至少其中之一係來自於複數資料變數4，因此與複數資料變數4相依的指令4可被該程式切片器1032所擷取出來。複數資料變數4的至少其中之一係來自於複數資料變數2, 3,或7，因此可知從節點121至節點123、節點122至節點123、節點126至節點123之資料流路徑至少分成3條路徑，其分別爲PA2, PB2, 以及PC2，從節點123至節點124之資料流路徑分別爲PA3, PB3, 以及PC3，且與複數資料變數2, 3, 7相依的指令2, 3, 7均被該程式切片器1032所擷取出來。同樣地，複數資料變數2的至少其中之一與複數資料變數3的至少其中之一均來自於複數資料變數1，因此往前追溯複數資料變數2與複數資料變數3的來源資料爲複數資料變數1的至少其中之一，而與該複數資料變數1相依的指令1可被該程式切片器1032所擷取出來，亦可知從節點120至節點121、節點120至節點122的資料流路徑分別爲PA1, PB1。複數資料變數4的至少其中之一亦源自於複數資料變數7，複數資料變數7的至少其中之一源自於複數資料變數6，故指令6, 7可被該程式切片器1032所擷取出來，節點125至節點126、節點126至節點123的資料流

路徑PC1, PC2亦可求得。藉由追蹤與資料變數相依的指令，所有的資料流路徑PA, PA, PC可求得，其中PA=PA1→PA2→PA3，PB=PB1→PB2→PB3，PC=PC1→PC2→PC3。指令1~7亦可被擷取到而作爲與安全性相關的指令群1010。另一種Forward Slicing的方法與Backward Slicing類似，只是往後追尋與往前追溯的不同而已。

【0024】 在另一較佳實施例中，資料流路徑的分析如下面反組譯後的組合語言程式碼：

【0025】 mov va, vb

add vc, va, vb

【0026】 追溯變數vc可知其由變數va與變數vb相加而得到，往回追溯變數va可知變數va是將變數vb移入變數va而得，因此源自於變數vb。故程式碼的資料流路徑爲vc←va←vb，且所使用到的指令群包含mov移動指令及add加法指令，指令群可以藉由將其廣義化來增加偵測惡意程式變種的偵測率。例如，add加法指令廣義化成爲包含算數邏輯的運算指令；若當上述惡意程式中的add加法指令被修改成爲sub減法指令、或是邏輯運匱指令而形成變種的惡意程式時，此變種的惡意程式將可被偵測到。

【0027】 在第二圖中，Backward Slicing的演算法(Algorithm)可由下列第一虛擬碼(Pseude Code 1)示意性地表示如下：

【0028】

Input:

Whole program CFG G=(V, E)

Critical point *ip*

Output:

Slicing graph $G_{(ip, B)}$=(V_B, E_B)

列1 R^S←TraceableSourceOperands(*ip*)

列2 *PATH*←AllSimplePathsEndAt(*ip*)

列3 V_B←V_B∪{*ip*}

列4

列5 **for** each simple path p∈*PATH* **do**

列6 *S*.push(R^S)

列7 **While** *S*≠Φ **do**

列8 *r*←*S*.pop()

列9 *u*←VertexOfferingRegister(*r*)

列10 **for** each vertex *v*∈Predecessors(*p*, *u*) **do**

列11 **if** *r*=TargetOperand(*v*) **then**

列12 V_B←V_B∪{*v*}

列13 E_B←E_B∪{(*u*, *v*)}

列14 *S*.push(TraceableSourceOperands(*v*))

列15 **break**

列16 **return** $G_{(ip,\ B)}$

【0029】 該第一虛擬碼的主要技術思想在於比較目前程式步驟的節點中的變數與上一個程式步驟的節點中的變數是否相同；若相同時則目前程式步驟的節點與上一個程式步驟的節點串連成資料流路徑，其說明如下。在該第一虛擬碼中的大寫英文字母*V, E, PATH, S,* R^S代表集合，下標大寫*B*代表追溯(Backward)，小寫英文字母*r*代表R^S集合的元素，小寫英文字母*p*代表*PATH*集合的元素，粗體英文字部分代表程式控制流程。輸入該語意模型擷取子系統103的是已知惡意程式樣本的所有程式之控制流程圖(Control Flow Graph, CFG)，其包含節點(*V*)與邊(*E*)，也就是第二圖中的步驟120~125

與資料流路徑PA, PB, PC。輸入該語意模型擷取子系統103的更包含已知惡意程式樣本的第i個關鍵點*ip*，也就是第i個指令。輸出的是經切片之資料流元素$G_{(ip,\ B)}$，在本發明較佳實施例第二圖中則是經切片之資料流元素G1, G2, G3。列1中的TraceableSourceOperands(*ip*)的功能是將已知惡意程式樣本102中由第*i*個指令所執行的所有資料變數取出，並將之儲存於具有堆疊*S*之暫存器R^S中，取出的順序可從程式流程中最後一個變數依序往前。列2的AllSimplePathsEndAt(*ip*)的功能係將已知惡意程式樣本102的單一資料流路徑取出，並將之儲存於*PATH*集合中，例如第二圖中的單一資料流路徑PA, PB, PC。列3中的節點集合V_B與第i個指令聯集後成爲新的節點集合V_B，從而第1~i個指令可被擷取到節點集合V_B。在列5中，對屬於路徑集合*PATH*中的每一個單一資料流路徑*p*重複執行列6~16的程式。在列6中，將暫存器R^S中的其中一個變數推進到堆疊集合*S*中。在列7中，當堆疊集合*S*中不是空集合時，重複執行列8~14的程式。在列8中，取出堆疊集合*S*中的變數至暫存器元素*r*。然後在列9中，藉由VertexOfferingRegister(*r*)的功能將暫存器元素*r*中的變數存至暫時的節點*u*。在列10中，對於任何程式的上一個步驟的節點屬於在*p*路徑上暫時的節點*u*時，重複執行列11~14的程式。在列11中，倘若程式上一個步驟的節點*v*的資料變數與程式目前步驟的節點*r*的資料變數相同時，則在列12中，程式上一個步驟的節點*v*加到程式目前步驟的節點的集合V_B中，且在列13中，程式上一個步驟的節點*v*與程式目前步驟的節點*r*之間的資料流路徑加入資料流路徑集合E_B中。然後在列14中，將程式上一個步驟的節點*v*之前一個步驟節點中的變數推進至堆疊集合*S*中。在列15中，中斷列10的程式for之迴圈，重覆上述列7~14的程式流程直到堆疊集合*S*中是空集合爲止才結束而可得到單一資料流路徑，然後再一次回到列5的程式for之迴圈以找出另一單獨的資料流路徑，如此一來可將程式切片分成複數個

單一資料流，如同第二圖中的資料流PA, PB, PC所示。在列16中的程式是將包含所有節點與邊的資料流路徑所構成的圖形傳回，而形成如第三圖中的資料流元素G1, G2, 或是G3。在第四圖中，已知惡意程式樣本102在經過關鍵點定位器1031收集與資料相依的指令群1010、經過程式切片器1032對已知惡意程式樣本102切片、經過該資料流元素產生器1033產生資料流元素G1, G2, 以及G3後，便完成了語意模型之擷取。同樣地，當輸入該語意模型擷取子系統爲該良性程式101時，則可得到如同第三圖中的資料流元素G1',以及G2'。

【0030】 列10~15的程式流程在找出單一資料流路徑，例如PA3→PA2→PA1。列6的程式則再取出另一資料變數，並找出另一條單一資料流路徑，例如PB3→PB2→PB1，以此類推，直到所有單一資料流路徑全部被找出爲止，然後將資料流元素$G_{(ip,\ B)}$輸出。若在第二圖中，複數資料變數1, 2, 3與複數資料變數6, 7之間互不相同，也就是其間沒有垂直連續的程式步驟之關係，這代表資料流路徑PA, PB, PC中的節點120, 121, 122與節點125, 126互爲獨立的，所以節點120, 121, 122與節點125, 126可分割成爲相互獨立的資料流元素G1, G2。在第二圖中，複數資料變數4中的至少其中之一個資料變數必定與複數資料變數2的至少其中之一個資料變數相同、複數資料變數4中的至少其中之一個資料變數必定與複數資料變數3的至少其中之一個資料變數相同、且複數資料變數4中的至少其中之一個資料變數必定與複數資料變數7的至少其中之一個資料變數相同，也就是其間有垂直連續的程式步驟之關係，而複數資料變數5與複數資料變數1, 2, 3之間互相獨立、複數資料變數5與複數資料變數6, 7之間互相獨立，故可以將節點123, 124形成資料流元素G3，其與資料流元素G1, G2相互獨立。在另一較佳實施例中，這些複數資料變數1~7也可以只是單一的資料變數。在第一圖中，輸入該語意模型

擷取子系統103的除了該已知惡意程式樣本以獲得資料流元素G1, G2, G3之外，良性程式101亦可輸入該語意模型擷取子系統103來得到資料流元素G1'與G2'，如同第三圖所示。

【0031】 在第二圖中，類似於Backward Slicing，Forward Slicing的演算法(Algorithm)可由下列第二虛擬碼(Pseude Code 2)示意性地表示如下：

【0032】

Input:

Whole program CFG G=(V, E)

Critical point ip

Output:

Slicing graph $G_{(ip, B)}$=(V_B, E_B)

列1 R^T←TraceableTargetOperand(ip)

列2 $PATH$←AllSimplePathsBeginAt(ip)

列3 V_F←V_F∪{ip}

列4

列5 **for** each simple path p∈$PATH$ **do**

列6 Q.push(R^T)

列7 **While** Q≠Φ **do**

列8 r←Q.pop()

列9 u←VertexOfferingRegister(r)

列10 **for** each vertex v∈Successors(p, u) **do**

列11 **if** r∈SourceOperands (v) **then**

列12 V_F←V_F∪{v}

列13 E_F←E_F∪{(u, v)}

列14　　Q.push(TraceableTargetOperand(v))

列15　　**else if** r=TargetOperand(v) **then**

列16　　**break**

列17　**return** $G_{(ip, B)}$

【0033】 Forward Slicing與Back Slicing相類似，因此只說明兩者之間不同的地方。在列15的程式中，倘若下一個節點的資料變數雖與上一個節點相同，但下一個節點中的指令之執行會將此相同的資料變數的數值改掉時，則不再追蹤下一個資料變數，因其不與下一個節點相關聯。在列16的程式中，中斷列10的程式之for迴圈而回到列7的程式之while迴圈，重覆上述列7~15的程式流程直到堆疊集合S中是空集合爲止才結束而可得到單一資料流路徑，然後再一次回到列5的程式for之迴圈以找出另一單獨的資料流路徑，如此一來可將程式切片分成複數個單一資料流，最後可得到資料流元素。

【0034】 在第三圖中示範了純化惡意程式的圖示，已知惡意程式樣本102以及良性程式101在經過語意模型擷取子系統103後，分別形成資料流元素G1, G2, G3以及資料流元素G1', G2'。在一較佳實施例中，將資料流元素G1與G1'比對、以及將資料流元素G3與G2'比對。比對後可知在已知惡意程式樣本102的資料流元素G1中與良性程式101的資料流元素G1'中的節點120, 122相同、節點120到122的資料流路徑(邊)相同以形成集合M1；在資料流元素G3中與資料流元素G1'中的節點124相同以形成集合M2，將已知惡意程式樣本102之集合去除集合M1與M2之聯集後，即可得到純化後的惡意程式107。

【0035】 在第三圖中，將已知惡意程式樣本102純化的演算法可由下列第三虛擬碼(Pseude Code 3)示意性地表示如下：

Input:Γ_S := 已知惡意程式樣本102的資料流元素之集合

Γ_B := 良性程式101的資料流元素之集合

Output: 純化後的惡意程式107

列1 **for** each components $C_S \in \Gamma_S$ **do**

列2 For each components $C_B \in \Gamma_B$ **do**

列3 $M \leftarrow$MaximunCommonSubgraph(C_S, C_B)

列4 *Synthe*$\leftarrow$*Synche*$\cup M$

列5 **if** NonTrivial (*Synthe*) **then**

列6 $C_S \leftarrow C_S$-*Synthe*

【0036】 該第三虛擬碼的說明如下，在列1中對於已知惡意程式樣本102的資料流元素之集合中的每個資料流元素重複執行列2~列6之程式。在列2中對於良性程式101的資料流元素之集合中的每個資料流元素重複執行列3~列6之程式。在列3中係取出C_S, C_B兩者之資料流元素中最大相同的部份。由於已知惡意程式樣本102的資料流元素以及良性程式101的資料流元素的數量可能是複數個，如第三圖的G1~G3以及G1'~G2'所示，G1與G1'比對時找到相同部分M1，將M1與一合成集合*Synthe*聯集，當G3與G2'比對時找到相同部分M2，將M2及合成集合*Synthe*聯集而形成一新的合成集合*Synthe*，如同列4所示。倘若新的合成集合*Synthe*是重要的，則將已知惡意程式樣本102的資料流元素之集合中去除該新的合成集合*Synthe*，最後可得到純化後的惡意程式107，如同列5~列6的程式之處理。

【0037】 請參考第五圖，其爲本發明待測程式106與純化惡意程式107相似度比較的示意圖。在一較佳實施例中，待測程式106在經過該語意模型擷取子系統103的處理後形成資料流元素X1, X2, X3, ..., Xn，純化惡意程式107在經過該語意模型擷取子系統103的處理後形成資料流元素Y1, Y2,

Y3, ..., Ym，資料流元素如前述爲各節點與連結各節點的邊所形成之圖形，因此資料流元素之間可進行圖形相似度比對。在第五圖中的惡意程式特徵比對子系統所採用的是二部圖匹配(bipartite match)來比對資料流元素X1, X2, X3, ..., Xn與資料流元素Y1, Y2, Y3, ..., Ym之相似度。

【0038】 在第五圖中，二部圖匹配比對時會針對待測程式106與純化惡意程式107中每個資料流元素進行相似度比對。例如，資料流元素X1會與資料流元素Y1, Y2, Y3, ..., Ym一一比對，資料流元素X2會與資料流元素Y1, Y2, Y3, ..., Ym一一比對，以下類推。最佳的匹配狀態是一對一相似的時候，例如資料流元素X1與資料流元素Y1相似、資料流元素X2與資料流元素Y2相似、資料流元素X3與資料流元素Y3相似、以及資料流元素Xn與資料流元素Ym相似的時候，則該惡意特徵比對子系統105判斷該待測程式106與該純化惡意程式107匹配，意即該待測程式106具備遂行惡意行爲。較差的匹配狀態是一對多相似的時候，例如資料流元素X1與資料流元素Y1, Y2, Y3, ..., Ym都相似，但資料流元素X2與資料流元素Y2、資料流元素X3與資料流元素Y3、以及資料流元素Xn與資料流元素Ym都不相似的時候，則該惡意特徵比對子系統105判斷該待測程式106與該純化惡意程式107不匹配，意即該待測程式106不具備遂行惡意行爲。待測程式106與純化惡意程式107的相似度可由其間之相似度距離來衡量，該相似度距離=(待測程式106∩純化惡意程式107)/(待測程式106∪純化惡意程式107)，當該相似度距離大於一門檻值時，則該惡意特徵比對子系統105判斷該待測程式106與該純化惡意程式107匹配。

【0039】 在第五圖中，待測程式106與純化惡意程式107相似度比較的演算法可由下列第四虛擬碼(Pseude Code 4)示意性地表示如下：

Input : Cn : 待測程式106的資料流元素集

Cm: 純化惡意程式的資料流元素集

Output : Cn以及Cm相似度的分數

列1 $Hn \leftarrow \Phi$; $Hm \leftarrow \Phi$

列2 **for** each vertex $v \in V$[Cn] **do**

列3 $Pv \leftarrow$ExtraAllTwoStepPaths(v)

列4 $Hn \leftarrow Hn \cup$djb2NgramHash(Pv)

列5 **for** each vertex $v \in V$[Cm] **do**

列6 $Pv \leftarrow$ExtraAllTwoStepPaths(v)

列7 $Hm \leftarrow Hm \cup$djb2NgramHash(Pv)

列8 $Score \leftarrow |Hn \cap Hm| / |Hn \cup Hm|$

【0040】 該第四虛擬碼的說明如下，在列1中，*Hn*爲待測程式每個節點至其相鄰兩節點的路徑之數值集，*Hm*爲純化惡意程式每個節點至其相鄰兩節點的路徑之數值集，*Hn*以及*Hm*一開始都被初始化設爲空集合。在列2~3中，對於在Cn中的每個節點，取出Cn中的每個節點至其相鄰兩節點的路徑到路徑集*Pv*。在列4中，將路徑集*Pv*內的路徑透過djb2NgramHash之雜湊功能轉換爲數值，直到所有路徑都轉換爲數值，此可提昇圖形比對速度。在列5~6中，對於在Cm中的每個節點，取出Cm中的每個節點至其相鄰兩節點的路徑到路徑集*Pv*。在列7中，將路徑集*Pv*內的路徑透過djb2NgramHash之雜湊功能轉換爲數值，直到所有路徑都轉換爲數值。最後在列8中，分數的判定可由公式$|Hn \cap Hm| / |Hn \cup Hm|$來判斷相似度的分數之高低，也就是相同路徑的個數除以所有路徑的個數。當分數超過一門檻值時，則判斷待測程式106與純化惡意程式107匹配。

【0041】 請參閱第六圖，其爲本發明在一系統10中產生一純化惡意程式107的方法的示意圖，該方法包含下列步驟：步驟S601，在一訓練階段使

一系統10接收一良性程式101以及一已知惡意程式樣本102。步驟S602，擷取在該良性程式101中與安全性相關的一指令群1010。步驟S603，透過複數資料流路徑PA, PB, PC來追蹤與該指令群1010相依的一程式片段111。步驟S604，將該程式片段111分割成複數獨立資料流元素G1', G2'。步驟S605，找出在該已知惡意程式樣本102中與該複數獨立資料流元素G1', G2'相同的一部分程式(M1∪M2)。步驟S606，從該已知惡意程式樣本102中去除該部分程式(M1∪M2)，以產生該純化惡意程式107。

【0042】 請同時參考第一圖、第五圖、以及第六圖，在第六圖中，該方法更包含下列步驟：在一偵測階段輸入一待測程式106至該系統10中。將該待測程式106分割成n個獨立資料流元素X1, X2, X3,...Xn，其中該n個獨立資料流元素的第n資料流元素包含一第n流程圖，該純化惡意程式107包含m個資料流元素Y1, Y2, Y3,..., Ym，該m個資料流元素的第m資料流元素包含一第m流程圖。將該第1~n流程圖與該第1~m流程圖進行相似度比對。將該第n流程圖中的所有資料路徑轉換成一第一組數值，並將該第m流程圖中的所有資料路徑轉換成一第二組數值，在該第n流程圖中的一第一資料路徑以在該第一組數值的一第一數值表示，在該第m流程圖中的一第二資料路徑以在該第二組數值的一第二數值表示，當該第一資料路徑與該第二資料路徑相同時，則該第一數值與該第二數值相同。找出在該第一組數值中與該第二組數值中相同數值的一第一個數，並找出在該第一組數值中與該第二組數值中不相同數值的一第二個數。將該第一個數除以該第一個數與該第二個數的和而得到一相似度估測值，當該相似度估測值大於或等於一特定門檻值時，則判定該第n流程圖與該第m流程圖相似。對該n個資料流元素與該m個資料流元素進行二部圖匹配(bipartite match)，當該n個流程圖與該m個流程圖一對一相似時，則判斷該待測程式106具備遂行惡意行為。

【0043】 請回到第一圖，待測程式106與模型資料庫108所提供的純化惡意程式107經由惡意特徵比對子系統105的相似度比對後，當待測程式106被判定具有遂行惡意行爲時，則該待測程式106可作爲新的惡意程式樣本存入模型資料庫108中，或是再送入惡意程式模型純化子系統104再進一步純化以提供至該模型資料庫108作爲新的惡意程式樣本。

【0044】 請參閱第七圖，其爲本發明判斷一待測程式106是否包含遂行惡意行爲的方法的示意圖，包含下列步驟：步驟S701，在一非惡意程式中擷取與安全性相關的指令群1010來建立一語意模型。去除在一已知惡意程式樣本102中與該語意模型相同的部分而形成一純化惡意程式107。比對該純化惡意程式107與該待測程式106，當兩者相似度達到一特定標準時，則判斷該待測程式106具備惡意行爲。

【0045】 請同時參考第五圖與第七圖，在一較佳實施例中，該待測程式106的該語意模型爲資料流元素X1, X2, X3, ..., Xn之集合，該非惡意程式可爲該良性程式101，該良性程式101的該語意模型爲資料流元素G1', G2'之集合，該純化惡意程式107的語意模型爲資料流元素Y1, Y2, Y3,..., Ym之集合。在步驟S701中建立該語意模型即前述的步驟S601、步驟S602、步驟S603、以及步驟S604所作的事情，步驟s702如同第三圖將已知惡意程式樣本102純化而形成該純化惡意程式107，步驟S703如同第五圖待測程式106與純化惡意程式107相似度比較，以判斷該待測程式106是否具備遂行惡意行爲。該特定標準包含一相似度估測值以及二部圖匹配(bipartite match)的一對一相似度，該相似度估測值例如爲該相似度距離=(待測程式106∩純化惡意程式107)/(待測程式106∪純化惡意程式107)，該一對一相似度例如以相似度的分數=｜$Hn \cap Hm$｜/｜$Hn \cup Hm$｜之標準來判斷。

【0046】 請參閱第八圖，其爲本發明判斷一待測程式106是否包含遂

行惡意行爲之指令的方法的示意圖，包含下列步驟：步驟S801，在一非惡意程式101中擷取與安全性相關的第一指令群1010來建立一語意模型G1', G2'。步驟S802，在一已知惡意程式樣本102中擷取與安全性相關的第二指令群，並去除該第二指令群中與該語意模型相同的部分而形成一純化惡意程式107。步驟S803，比對該純化惡意程式107與該待測程式106中與安全性相關的指令群，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行爲之指令。

【0047】 請參閱第九圖，其爲本發明另一較佳實施例判斷一待測程式106是否包含遂行惡意行爲之指令的方法的示意圖，包含下列步驟：步驟S901，爲一非惡意程式101建立一語意模型。步驟S902，去除在一已知惡意程式樣本102中與該語意模型相同的部分而形成一純化惡意程式107。步驟S903，比對該純化惡意程式107與該待測程式106，當兩者相似度達到一特定標準時，則判斷該待測程式106包含遂行惡意行爲之指令。

【0048】 請參閱表1，下列表1列出了已知惡意程式樣本102在純化之前資料流元素的數量，以及純化後的惡意程式的資料流的數量。從表1可知已知惡意程式樣本102在經過純化後的資料流元素之數量減少，節點數量也隨之減少，此可達到在偵測惡意程式時降低誤報率的功效。

表 1

惡意程式樣本	資料流元素數量	純化後的惡意程式	
		資料流元素數量	節點數量
DroidKungFu	9	1	6
AnserverBot	4	2	5
BaseBridge	6	1	15
Geinimi	5	3	11

Pjapps	5	1	7
GoldDream	4	1	8
DroidDreamLight	5	1	4
ADRD	4	1	4
DroidDream	6	1	7
jSMSHider	9	2	15
Zsone	2	1	2
Bgserv	7	2	13
BeanBot	5	1	10
GingerMaster	4	1	16
HippoSMS	5	1	6

【0049】 請參閱表2，下列表2列出了偵測惡意程式的偵測率1、偵測率2、以及偵測率3，偵測率1是該系統10第一次偵測該待測程式106時的偵測率，該待測程式106經過系統10的處理後會產生新的惡意程式樣本，該新的惡意程式樣本可輸入該系統10以進行第二次偵測，同樣的方式可再次進行第三次偵測。最佳狀況是惡意程式樣本102與良性程式101相同的部分較多，因而純化惡意程式的功能較爲顯著，在第一次偵測時其偵測率達100%，例如jSMSHider, Zsone等惡意程式。總體而言，在第二次偵測時的偵測率2會比第一次偵測時的偵測率高，在第三次偵測時的偵測率3會比第二次偵測時的偵測率2高，例如在偵測Pjapps惡意程式樣本時，偵測率3＞偵測率2＞偵測率1，或是平均第三次總偵測率0.927＞平均第二次總偵測率0.911＞平均第一次總偵測率0.839。當惡意程式樣本102與良性程式101相同的部分較少時，純化惡意程式的功能較不顯著，偵測率也會不理想，例如HippoSMS, DroidKungFu等惡意程式。

表2

惡意程式樣本	樣本數量	偵測率1	偵測率2	偵測率3
DroidKungFu	472	0.76	0.76	0.76
AnserverBot	186	0.99	0.99	0.99
BaseBridge	121	0.36	0.74	0.74
Geinimi	68	0.74	1	1
Pjapps	57	0.46	0.58	0.81
GoldDream	46	0.67	1	1
DroidDreamLight	45	1	1	1
ADRD	21	1	1	1
DroidDream	15	0.93	0.93	0.93
jSMSHider	15	1	1	1
Zsone	11	1	1	1
Bgserv	8	1	1	1
BeanBot	7	1	1	1
GingerMaster	3	1	1	1
HippoSMS	3	0.67	0.67	0.67
總計與平均	1078	0.839	0.911	0.927

【0050】 實施例

【0051】 1. 一種在一系統中產生一純化惡意程式的方法，該方法包含下列步驟：在一訓練階段使該系統接收一良性程式以及一已知惡意程式樣本。擷取在該良性程式中與安全性相關的一指令群。透過複數資料流路徑來追蹤與該指令群相依的一程式片段。將該程式片段分割成複數獨立資

料流元素。找出在該已知惡意程式樣本中與該複數獨立資料流元素相同的一部分程式。從該已知惡意程式樣本中去除該部分程式，以產生該純化惡意程式。

【0052】 2. 如實施例1所述的方法，更包含下列步驟：比對該複數獨立資料流元素和該已知惡意程式樣本，以找出該部分程式。對該良性程式的位元組碼進行反組譯以找出該指令群，其中該指令群包含一隱私相關的指令群、一未受Dalvik虛擬機器保護的指令群、及一背景自動執行的指令群。針對該複數資料流路徑中的每一個資料單元往前追溯其來源資料，並找出執行該來源資料的所有路徑，或針對該複數資料流路徑中的每一個資料單元往後查找其後續執行的資料，並找出執行該後續執行的資料的所有路徑，以獲得該程式片段。將該指令群的語意廣義化。

【0053】 3. 如實施例1-2所述的方法，更包含下列步驟：在一偵測階段輸入一待測程式至該系統中。將該待測程式分割成n個獨立資料流元素，其中該n個獨立資料流元素的第n資料流元素包含一第n流程圖，該純化惡意程式包含m個資料流元素，該m個資料流元素的第m資料流元素包含一第m流程圖。將該第1~n流程圖與該第1~m流程圖進行相似度比對。將該第n流程圖中的所有資料路徑轉換成一第一組數值，並將該第m流程圖中的所有資料路徑轉換成一第二組數值，在該第n流程圖中的一第一資料路徑以在該第一組數值的一第一數值表示，在該第m流程圖中的一第二資料路徑以在該第二組數值的一第二數值表示，當該第一資料路徑與該第二資料路徑相同時，則該第一數值與該第二數值相同。找出在該第一組數值中與該第二組數值中相同數值的一第一個數，並找出在該第一組數值中與該第二組數值中不相同數值的一第二個數。將該第一個數除以該第一個數與該第二個數的和而得到一相似度估測值，當該相似度估測值大於或等於一特定門檻值時，

則判定該第n流程圖與該第m流程圖相似。對該n個資料流元素與該m個資料流元素進行二部圖匹配(bipartite match)，當該n個流程圖與該m個流程圖一對一相似時，則判斷該待測程式具備遂行惡意行為。

【0054】 4. 一種判斷一待測程式是否包含一惡意程式的系統，該系統接收一良性程式與一已知惡意程式樣本，該系統包含一語意模型擷取子系統、一惡意程式模型純化子系統、以及一惡意特徵比對子系統。該語意模型擷取子系統擷取該良性程式中與安全性相關的一指令群、透過複數資料流路徑來追蹤與該指令群相依的一程式片段、以及將該程式片段分割成複數獨立資料流元素。該惡意程式模型純化子系統將與該複數獨立資料流元素相同的部分從該已知惡意程式樣本中去除，以形成一純化惡意程式。該惡意特徵比對子系統比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式包含該惡意程式。

【0055】 5. 如實施例4所述的系統，其中該語意模型擷取子系統包含一安全相關指令擷取器、一程式分割器、以及一資料流元素產生器(footprint generator)。該安全相關指令擷取器對該良性程式的位元組碼進行反組譯以找出該指令群，其中該指令群包含一隱私相關的指令群、一未受Dalvik虛擬機器保護的指令群、及一背景自動執行的指令群。該程式分割器對該複數資料流路徑中的每一個資料單元往前追溯其來源資料，並找出執行該來源資料的所有路徑，或針對該複數資料流路徑中的每一個資料單元往後查找其後續執行的資料，並找出執行該後續執行的資料的所有路徑，以獲得該程式片段，然後該程式分割器將該程式片段分割成複數獨立資料流元素。該資料流元素產生器(footprint generator)將該指令群的語意廣義化，並輸出複數獨立資料流元素。該系統在一訓練階段接收該良性程式，且在一偵測階段接收該待測程式。該語意模型擷取子系統中的一程式分割器將該待測

程式分割成的n個獨立資料流元素，其中該n個獨立資料流元素的第n資料流元素包含一第n流程圖，該純化惡意程式包含m個資料流元素，該m個資料流元素的第m資料流元素包含一第m流程圖。該惡意特徵比對子系統將該第1~n流程圖與該第1~m流程圖進行相似度比對、將該第N流程圖中的所有資料路徑轉換成一第一組數值，並將該第m流程圖中的所有資料路徑轉換成一第二組數值，在該第n流程圖中的一第一資料路徑以在該第一組數值的一第一數值表示，在該第m流程圖中的一第二資料路徑以在該第二組數值的一第二數值表示，當該第一資料路徑與該第二資料路徑相同時，則該第一數值與該第二數值相同。該惡意特徵比對子系統找出在該第一組數值中與該第二組數值中相同數值的一第一個數、找出在該第一組數值中與該第二組數值中不相同數值的一第二個數、將該第一個數除以該第一個數與該第二個數的和而得到一相似度估測值，當該相似度估測值大於或等於一特定門檻值時，則判定該第n流程圖與該第m流程圖相似。該惡意特徵比對子系統對該n個資料流元素與該m個資料流元素進行二部圖匹配(bipartite match)，當該第n個流程圖與該第m個流程圖一對一相似時，則判斷該待測程式具備遂行惡意行爲。

【0056】 6. 一種判斷一待測程式是否包含遂行惡意行爲的方法，包含下列步驟：在一非惡意程式中擷取與安全性相關的指令群來建立一語意模型。去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式。比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式具備惡意行爲。

【0057】 7. 如實施例6所述的方法，更包含下列步驟：透過複數資料流路徑來追蹤與該指令群相依的一程式片段。針對該複數資料流路徑中的每一個資料單元往前追溯其來源資料，並找出執行該來源資料的所有路

徑，或針對該複數資料流路徑中的每一個資料單元往後查找其後續執行的資料，並找出執行該後續執行的資料的所有路徑，以獲得該程式片段。將該程式片段分割成複數獨立資料流元素。將該指令群的語意廣義化。比對該複數資料流元素與該已知惡意程式樣本，將與該複數資料流元素相同的部分從該已知惡意程式樣本中去除，以形成該純化惡意程式。在一偵測階段輸入該待測程式至該系統中。將該待測程式分割成n個獨立資料流元素，其中該n個獨立資料流元素的第n資料流元素包含一第n流程圖，該純化惡意程式包含m個資料流元素，該m個資料流元素的第m資料流元素包含一第m流程圖。將該第1~n流程圖與該第1~m流程圖進行相似度比對。將該第n流程圖中的所有資料路徑轉換成一第一組數值，並將該第m流程圖中的所有資料路徑轉換成一第二組數值，在該第n流程圖中的一第一資料路徑以在該第一組數值的一第一數值表示，在該第m流程圖中的一第二資料路徑以在該第二組數值的一第二數值表示，當該第一資料路徑與該第二資料路徑相同時，則該第一數值與該第二數值相同。找出在該第一組數值中與該第二組數值中相同數值的一第一個數，並找出在該第一組數值中與該第二組數值中不相同數值的一第二個數。將該第一個數除以該第一個數與該第二個數的和而得到一相似度估測值，當該相似度估測值大於或等於一特定門檻值時，則判定該第n流程圖與該第m流程圖相似。對該n個資料流元素與該m個資料流元素進行二部圖匹配(bipartite match)，當該n個流程圖與該m個流程圖一對一相似時，則判斷該待測程式具備惡意行為。

【0058】 8. 如實施例6-7所述的方法，其中該純化惡意程式作為一新的惡意程式樣本以輸入該系統。該特定標準包含一相似度估測值以及二部圖匹配(bipartite match)的一對一相似度。對該無惡意程式的位元組碼進行反組譯以找出該指令群，其中該指令群包含一隱私相關的指令群、一未受

Dalvik虛擬機器保護的指令群、及一背景自動執行的指令群。

【0059】 9. 一種判斷一待測程式是否包含遂行惡意行爲之指令的方法，包含下列步驟：在一非惡意程式中擷取與安全性相關的第一指令群來建立一語意模型。在一已知惡意程式樣本中擷取與安全性相關的第二指令群，並去除該第二指令群中與該語意模型相同的部分而形成一純化惡意程式。比對該純化惡意程式與該待測程式中與安全性相關的指令群，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行爲之指令。

【0060】 10. 一種判斷一待測程式是否包含遂行惡意行爲之指令的方法，包含下列步驟：爲一非惡意程式建立一語意模型。去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式。比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行爲之指令。

【0061】 綜上所述，本發明的說明與實施例已揭露於上，然其非用來限制本發明，凡習知此技藝者，在不脫離本發明的精神與範圍之下，當可做各種更動與修飾，其仍應屬在本發明專利的涵蓋範圍之內。

【符號說明】

【0062】

10:判斷一待測程式是否包含一惡意程式的系統	120, 121, 122, 123, 124, 125, 126：節點
102：已知惡意程式樣本	101：良性程式
104：惡意程式模型純化子系統	103：語意模型擷取子系統
106：待測程式	105：惡意特徵比對子系統
108：模型資料庫	107：純化惡意程式
110：待測資料集	1010：指令群

111：程式片段　　109：訓練資料集

G1, G2, G3, G1', G2', X1~Xn, Y1~Ym：資料流元素　　PA1, PA2, PA3, PA, PB1, PB2, PB3, PB, PC1, PC2, PC3, PC：資料流路徑

M1, M2：良性程式與已知惡意程式的相同的部份　　1031 ：關鍵點定位器

1032：程式切片器　　1033：資料流元素產生器

【生物材料寄存】

國內寄存資訊【請依寄存機構、日期、號碼順序註記】

國外寄存資訊【請依寄存國家、機構、日期、號碼順序註記】

【序列表】(請換頁單獨記載)

申請專利範圍

1. 一種在一系統中產生一純化惡意程式的方法，該方法包含下列步驟：

在一訓練階段使該系統接收一良性程式以及一已知惡意程式樣本；

擷取在該良性程式中與安全性相關的一指令群；

透過複數資料流路徑來追蹤與該指令群相依的一程式片段；

將該程式片段分割成複數獨立資料流元素；

找出在該已知惡意程式樣本中與該複數獨立資料流元素相同的一部分程式；以及

從該已知惡意程式樣本中去除該部分程式，以產生該純化惡意程式。

2. 如申請專利範圍第1項所述的方法，更包含下列步驟：

比對該複數獨立資料流元素和該已知惡意程式樣本，以找出該部分程式;

對該良性程式的位元組碼進行反組譯以找出該指令群，其中該指令群包含一隱私相關的指令群、一未受Dalvik虛擬機器保護的指令群、及一背景自動執行的指令群；

針對該複數資料流路徑中的每一個資料單元往前追溯其來源資料，並找出執行該來源資料的所有路徑，或針對該複數資料流路徑中的每一個資料單元往後查找其後續執行的資料，並找出執行該後續執行的資料的所有路徑，以獲得該程式片段；以及

將該指令群的語意廣義化。

3. 如申請專利範圍第1項所述的方法，更包含下列步驟：

在一偵測階段輸入一待測程式至該系統中；

將該待測程式分割成n個獨立資料流元素，其中該n個獨立資料流元素的第n資料流元素包含一第n流程圖，該純化惡意程式包含m個資料流元素，該m個資料流元素的第m資料流元素包含一第m流程圖；

將該第1~n流程圖與該第1~m流程圖進行相似度比對；

將該第n流程圖中的所有資料路徑轉換成一第一組數值，並將該第m流程圖中的所有資料路徑轉換成一第二組數值，在該第n流程圖中的一第一資料路徑以在該第一組數值的一第一數值表示，在該第m流程圖中的一第二資料路徑以在該第二組數值的一第二數值表示，當該第一資料路徑與該第二資料路徑相同時，則該第一數值與該第二數值相同；

找出在該第一組數值中與該第二組數值中相同數值的一第一個數，並找出在該第一組數值中與該第二組數值中不相同數值的一第二個數；

將該第一個數除以該第一個數與該第二個數的和而得到一相似度估測值，當該相似度估測值大於或等於一特定門檻值時，則判定該第n流程圖與該第m流程圖相似；以及

對該n個資料流元素與該m個資料流元素進行二部圖匹配(bipartite match)，當該n個流程圖與該m個流程圖一對一相似時，則判斷該待測程式具備遂行惡意行為。

4. 一種判斷一待測程式是否包含一惡意程式的系統，該系統接收一良性程式與一已知惡意程式樣本，該系統包含：

一語意模型擷取子系統，擷取該良性程式中與安全性相關的一指令群、透過複數資料流路徑來追蹤與該指令群相依的一程式片段、以及將該程式片段分割成複數獨立資料流元素；

一惡意程式模型純化子系統，將與該複數獨立資料流元素相同的部分

從該已知惡意程式樣本中去除，以形成一純化惡意程式；以及

一惡意特徵比對子系統，比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式包含該惡意程式。

5. 如申請專利範圍第4項所述的系統，其中：

該語意模型擷取子系統包含：

一安全相關指令擷取器，對該良性程式的位元組碼進行反組譯以找出該指令群，其中該指令群包含一隱私相關的指令群、一未受Dalvik虛擬機器保護的指令群、及一背景自動執行的指令群；

一程式分割器，對該複數資料流路徑中的每一個資料單元往前追溯其來源資料，並找出執行該來源資料的所有路徑，或針對該複數資料流路徑中的每一個資料單元往後查找其後續執行的資料，並找出執行該後續執行的資料的所有路徑，以獲得該程式片段，然後該程式分割器將該程式片段分割成複數獨立資料流元素；以及

一資料流元素產生器(footprint generator)，將該指令群的語意廣義化，並輸出複數獨立資料流元素；

該系統在一訓練階段接收該良性程式，且在一偵測階段接收該待測程式；

該語意模型擷取子系統中的一程式分割器將該待測程式分割成的n個獨立資料流元素，其中該n個獨立資料流元素的第n資料流元素包含一第n流程圖，該純化惡意程式包含m個資料流元素，該m個資料流元素的第m資料流元素包含一第m流程圖；

該惡意特徵比對子系統將該第1~n流程圖與該第1~m流程圖進行相似度比對、將該第N流程圖中的所有資料路徑轉換成一第一組數值，並將該第m

流程圖中的所有資料路徑轉換成一第二組數值，在該第n流程圖中的一第一資料路徑以在該第一組數值的一第一數值表示，在該第m流程圖中的一第二資料路徑以在該第二組數值的一第二數值表示，當該第一資料路徑與該第二資料路徑相同時，則該第一數值與該第二數值相同；

該惡意特徵比對子系統找出在該第一組數值中與該第二組數值中相同數值的一第一個數、找出在該第一組數值中與該第二組數值中不相同數值的一第二個數、將該第一個數除以該第一個數與該第二個數的和而得到一相似度估測值，當該相似度估測值大於或等於一特定門檻值時，則判定該第n流程圖與該第m流程圖相似；以及

該惡意特徵比對子系統對該n個資料流元素與該m個資料流元素進行二部圖匹配(bipartite match)，當該第n個流程圖與該第m個流程圖一對一相似時，則判斷該待測程式具備遂行惡意行為。

6. 一種判斷一待測程式是否包含遂行惡意行為的方法，包含下列步驟：

在一非惡意程式中擷取與安全性相關的指令群來建立一語意模型；

去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式；以及

比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式具備惡意行為。

7. 如申請專利範圍第6項所述的方法，更包含下列步驟：

透過複數資料流路徑來追蹤與該指令群相依的一程式片段；

針對該複數資料流路徑中的每一個資料單元往前追溯其來源資料，並找出執行該來源資料的所有路徑，或針對該複數資料流路徑中的每一個資

料單元往後查找其後續執行的資料，並找出執行該後續執行的資料的所有路徑，以獲得該程式片段；

將該程式片段分割成複數獨立資料流元素；

將該指令群的語意廣義化；

比對該複數資料流元素與該已知惡意程式樣本，將與該複數資料流元素相同的部分從該已知惡意程式樣本中去除，以形成該純化惡意程式；

在一偵測階段輸入該待測程式至該系統中；

將該待測程式分割成n個獨立資料流元素，其中該n個獨立資料流元素的第N資料流元素包含一第n流程圖，該純化惡意程式包含m個資料流元素，該m個資料流元素的第m資料流元素包含一第m流程圖；

將該第1~n流程圖與該第1~m流程圖進行相似度比對；

將該第n流程圖中的所有資料路徑轉換成一第一組數值，並將該第m流程圖中的所有資料路徑轉換成一第二組數值，在該第n流程圖中的一第一資料路徑以在該第一組數值的一第一數值表示，在該第m流程圖中的一第二資料路徑以在該第二組數值的一第二數值表示，當該第一資料路徑與該第二資料路徑相同時，則該第一數值與該第二數值相同；

找出在該第一組數值中與該第二組數值中相同數值的一第一個數，並找出在該第一組數值中與該第二組數值中不相同數值的一第二個數；

將該第一個數除以該第一個數與該第二個數的和而得到一相似度估測值，當該相似度估測值大於或等於一特定門檻值時，則判定該第n流程圖與該第m流程圖相似；以及

對該n個資料流元素與該m個資料流元素進行二部圖匹配(bipartite match)，當該n個流程圖與該m個流程圖一對一相似時，則判斷該待測程式具備惡意行為。

5

8. 如申請專利範圍第6項所述的方法，其中：

該純化惡意程式作爲一新的惡意程式樣本以輸入該系統；

該特定標準包含一相似度估測值以及二部圖匹配(bipartite match)的一對一相似度；以及

對該無惡意程式的位元組碼進行反組譯以找出該指令群，其中該指令群包含一隱私相關的指令群、一未受Dalvik虛擬機器保護的指令群、及一背景自動執行的指令群。

9. 一種判斷一待測程式是否包含遂行惡意行爲之指令的方法，包含下列步驟：

在一非惡意程式中擷取與安全性相關的第一指令群來建立一語意模型；

在一已知惡意程式樣本中擷取與安全性相關的第二指令群，並去除該第二指令群中與該語意模型相同的部分而形成一純化惡意程式；以及

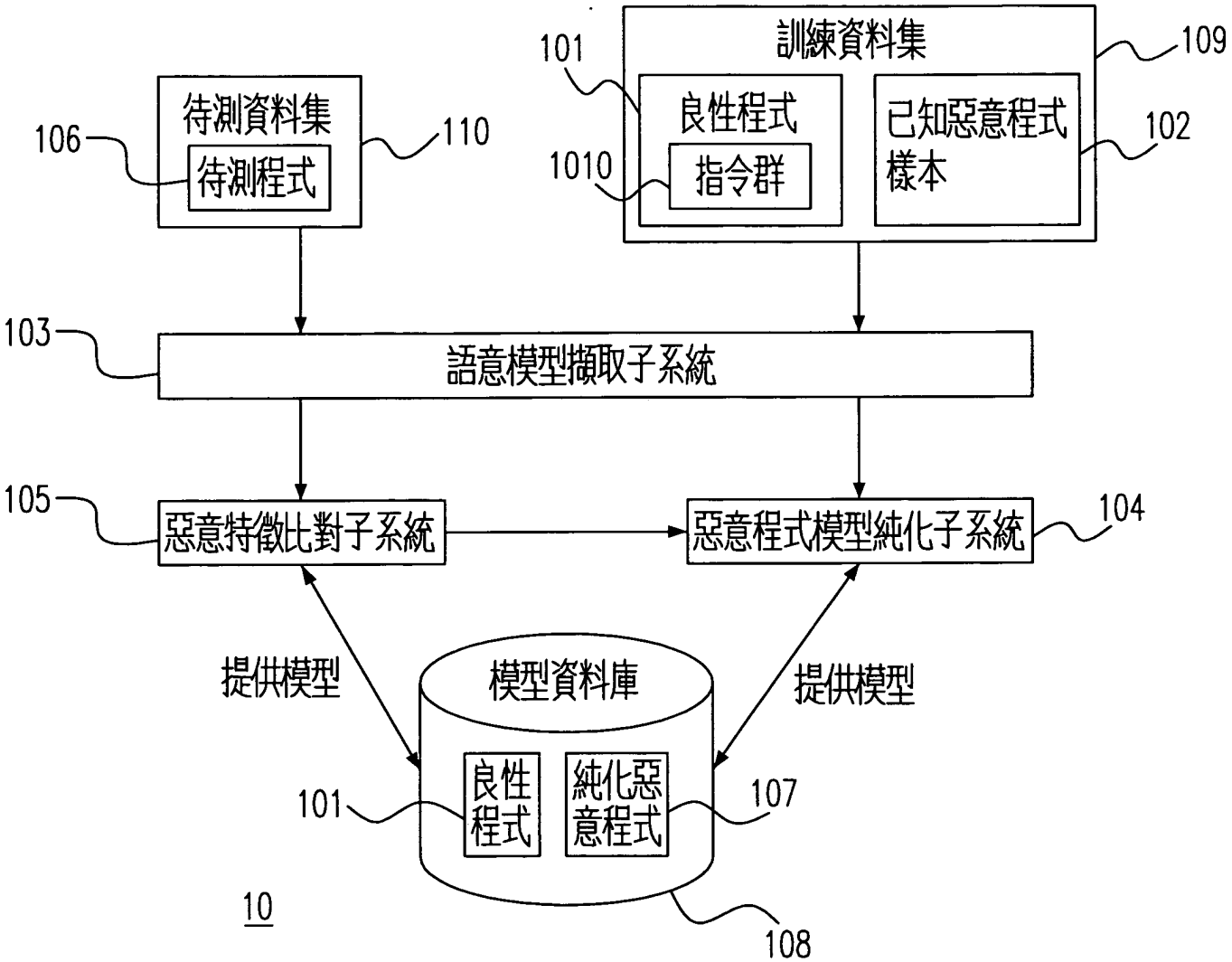
比對該純化惡意程式與該待測程式中與安全性相關的指令群，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行爲之指令。

10. 一種判斷一待測程式是否包含遂行惡意行爲之指令的方法，包含下列步驟：

爲一非惡意程式建立一語意模型；

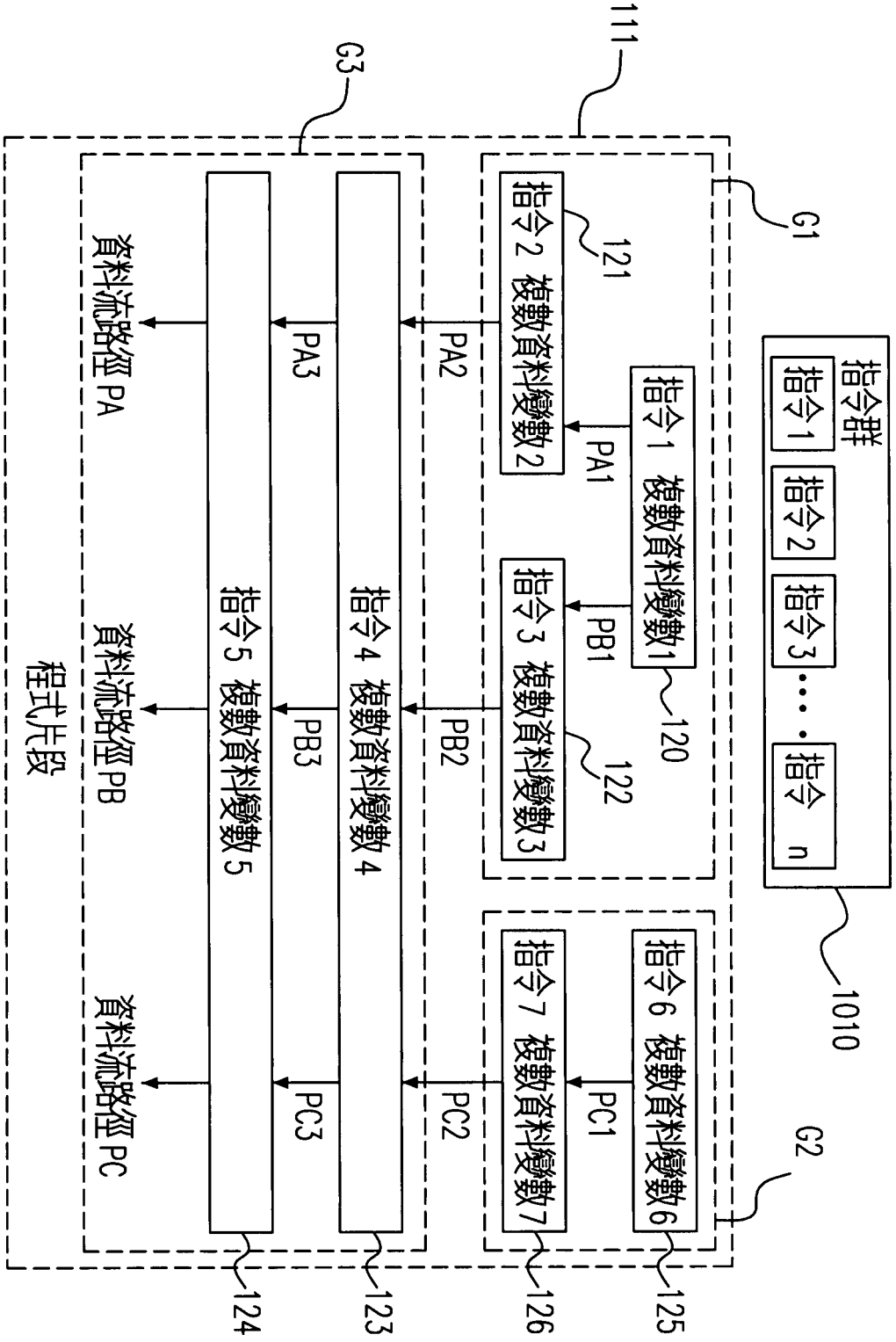
去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式；以及

比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行爲之指令。

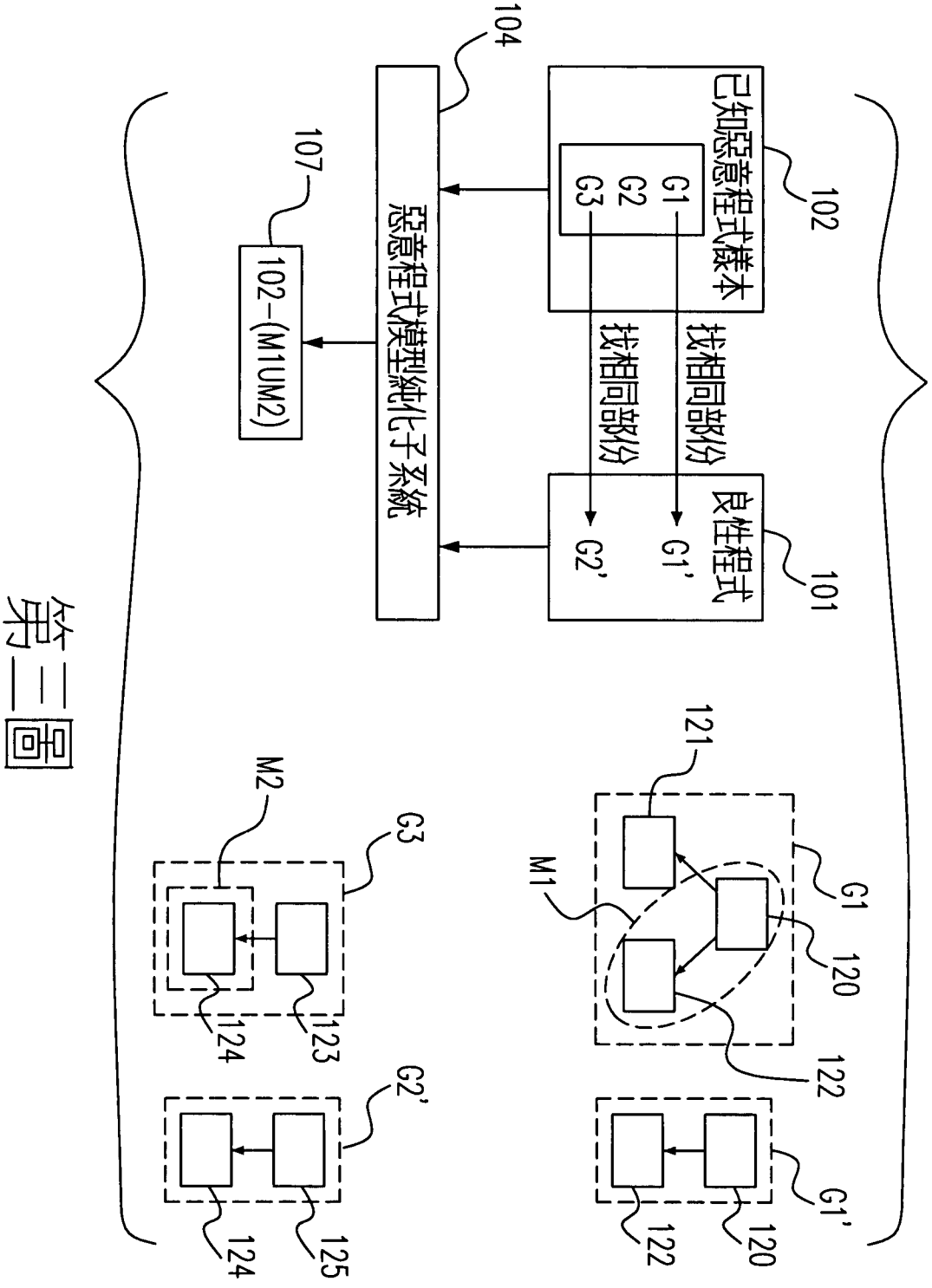


第一圖

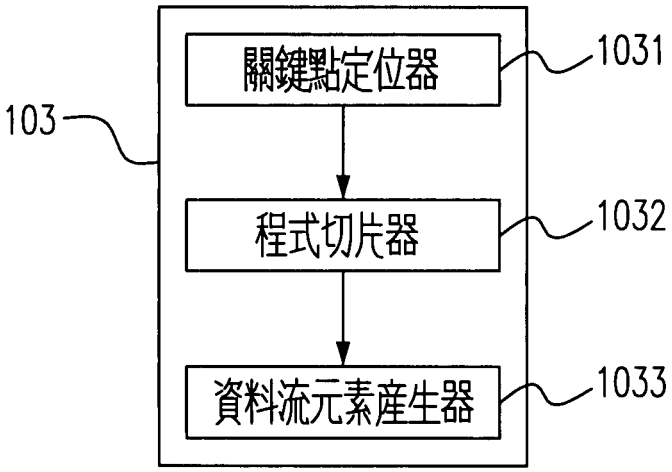
201508534


指令群
指令1
指令2
指令3
指令 n
1010
111
G1
G2
G3
指令1 複數資料變數1
120
PA1
PB1
指令2 複數資料變數2
121
指令3 複數資料變數3
122
PA2
PB2
指令6 複數資料變數6
125
PC1
PC2
指令7 複數資料變數7
126
指令4 複數資料變數4
123
PA3
PB3
PC3
指令5 複數資料變數5
124
資料流路徑 PA
資料流路徑 PB
資料流路徑 PC
程式片段

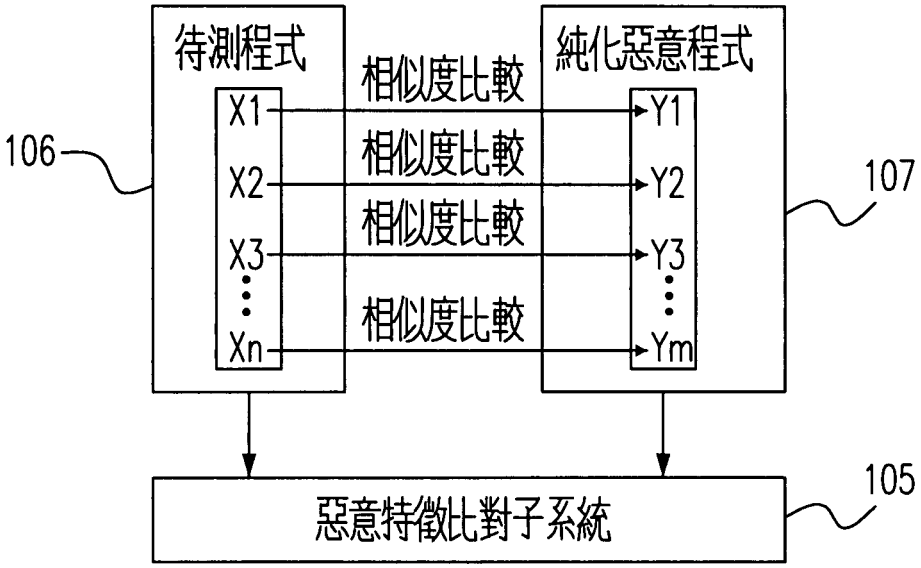
第一圖


102
已知惡意程式樣本
G1
G2
G3
找相同部份
找相同部份
101
良性程式
G1'
G2'
104
惡意程式模型純化子系統
107
102-(M1UM2)
121
M1
G1
120
122
G1'
120
122
M2
G3
123
124
G2'
125
124

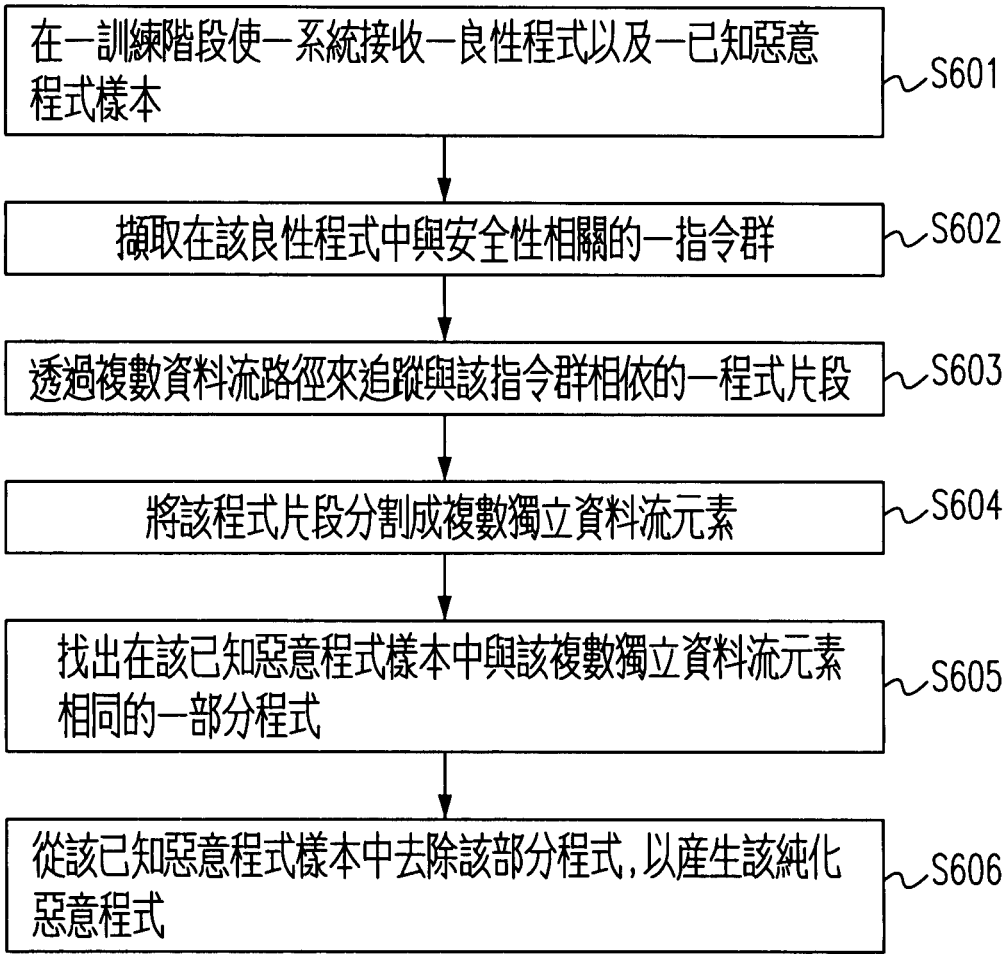
第二圖


關鍵點定位器
1031
103
程式切片器
1032
資料流元素產生器
1033

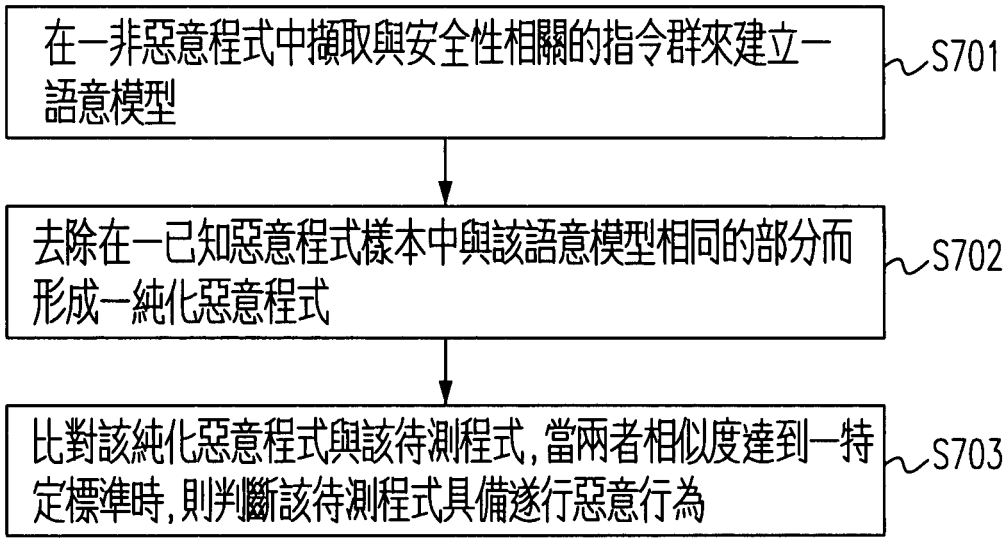
第四圖


待測程式
純化惡意程式
相似度比較
X1
X2
X3
Xn
Y1
Y2
Y3
Ym
106
107
惡意特徵比對子系統
105

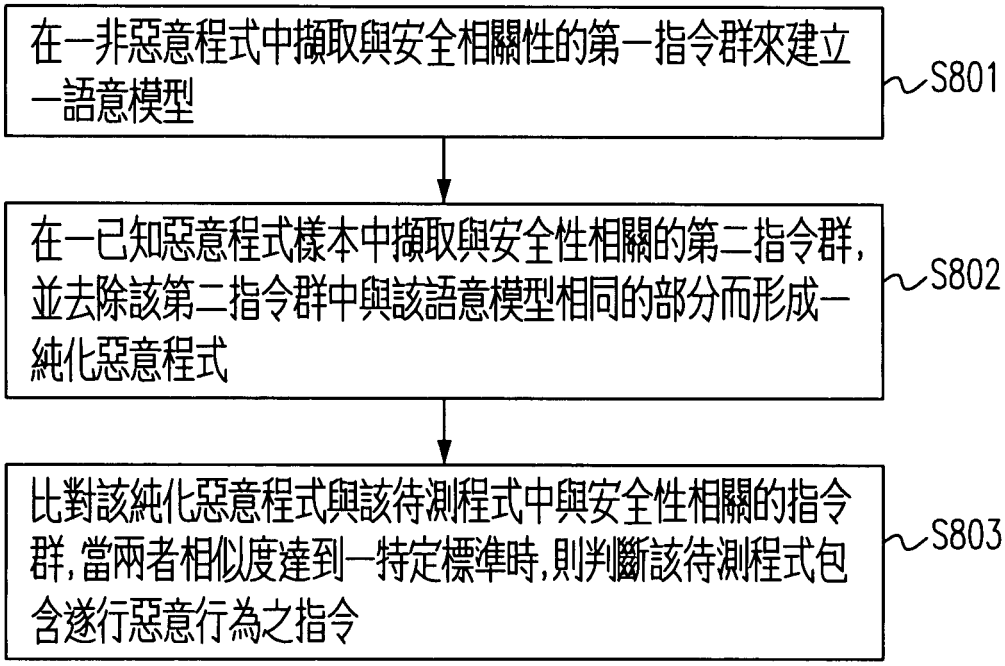
第五圖


在一訓練階段使一系統接收一良性程式以及一已知惡意程式樣本
S601
擷取在該良性程式中與安全性相關的一指令群
S602
透過複數資料流路徑來追蹤與該指令群相依的一程式片段
S603
將該程式片段分割成複數獨立資料流元素
S604
找出在該已知惡意程式樣本中與該複數獨立資料流元素相同的一部分程式
S605
從該已知惡意程式樣本中去除該部分程式，以產生該純化惡意程式
S606

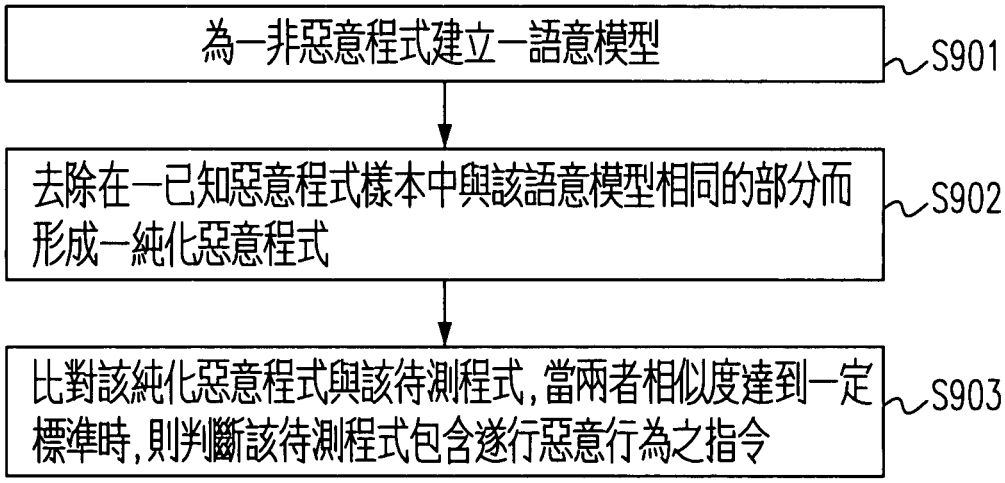
第六圖


在一非惡意程式中擷取與安全性相關的指令群來建立一語意模型
S701
去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式
S702
比對該純化惡意程式與該待測程式，當兩者相似度達到一特定標準時，則判斷該待測程式具備遂行惡意行為
S703

第七圖


在一非惡意程式中擷取與安全相關性的第一指令群來建立一語意模型
S801
在一已知惡意程式樣本中擷取與安全性相關的第二指令群，並去除該第二指令群中與該語意模型相同的部分而形成一純化惡意程式
S802
比對該純化惡意程式與該待測程式中與安全性相關的指令群，當兩者相似度達到一特定標準時，則判斷該待測程式包含遂行惡意行為之指令
S803

第八圖


為一非惡意程式建立一語意模型
S901
去除在一已知惡意程式樣本中與該語意模型相同的部分而形成一純化惡意程式
S902
比對該純化惡意程式與該待測程式，當兩者相似度達到一定標準時，則判斷該待測程式包含遂行惡意行為之指令
S903

第九圖