## 【論文題目】

比較Sigma規則及各深度學習模型在系統日誌上偵測 MITRE Techniques 的成效

（比較理想的題目名稱：基於XX方法在系統日誌進行 APT 偵測？）

## 【研究動機與介紹】

進階持續性威脅 (Advanced Persistent Threats, APTs) 是由經驗豐富的攻擊者針對精心挑選的目標發起的網路攻擊，這些目標可能包括政府、核心基礎設施（例如能源、交通、通訊）以及重要產業（例如軍事、金融、醫療）。APTs 可能造成嚴重的安全威脅，如竊取敏感資料和損害系統完整性，案例如：APT40 對一帶一路國家竊取海軍相關機密科技、BlackEnergy 惡意軟體攻擊烏克蘭電力網、Equifax 中國解放軍盜走1.5億名美國民眾個資等。APTs 所帶來的安全威脅已成為當今數位世界中極為嚴重且嚴峻的挑戰。面對這樣的威脅，有效的APTs偵測成為了保護數據和系統安全的關鍵環節。

觀察過往的研究文獻，溯源資料（Provenance Data）是 APT 偵測任務的最佳來源[1][2][3]。溯源資料通常是從系統審核日誌（Audit Log）或是整理而來，而這些系統審核日誌多是藉由 Linux Auditd, Windows ETW 等工具收集而來。大部分的研究[4][5][6][7][8]會將溯源資料建模為有向無環圖（Directed Acyclic Graph），並稱其為溯源圖（Provenance Graph），圖中節點的部分代表系統資源的實體，如：程序（Process）、檔案（File）、登錄檔（Registry），邊代表系統系統動作，如：建立程序（Process Create）、寫檔案（File Write）、取得登錄鍵（Registry Key Query）。

大部分現有基於溯源圖的 APT 偵測系統（如：2018 SLEUTH[4] / 2019 Holmes[5] / 2020 Rapsheet[6] / 2022 SteinerLog[7] / 2023 APTSHIELD[8]）都是根據威脅知識庫（如：Kill Chain Model[9], ATT&CK Model [10]）去人工設計各種規則，儘管這些基於規則的 APT 偵測方式非常有效且被廣泛引用，但它們仍然存在以下局限性：

* 第一，規則的設計是一項極其困難的任務，因為它需要威脅知識庫、作業系統和計算機網絡的專業領域知識。
* 第二，規則缺乏泛化能力，因為它們是在特定條件（例如攻擊策略、網絡環境）下設計的。 當這些特定條件發生變化時，規則很容易過時並失效，必須進行調整甚至重新設計。
* 第三，規則僅限於人類經驗，難以處理潛在或未被觀察到的攻擊模式。
* 第四，規則多是使用 Regular Expression 或是有特定語法結構，難以處理複雜的攻擊模式。

另一種 APT 偵測方式是基於深度學習方法的偵測模型，與基於規則的 APT 偵測方式相比，基於深度學習方法的 APT 偵測方式有以下好處：

* 讓電腦自動從訓練樣本中創建偵測模型，進而達到不需仰賴專業領域知識，就能建立偵測模型的目的。
* 透過自動化重新訓練或是調整參數的方式，偵測模型可以快速地適應新的變化。

但相反的，基於深度學習的 APT 偵測也面臨許多挑戰：

* 對一個深度學習的模型來說，輸入是低階的系統事件日誌，而輸出卻是具備攻擊行為模式的的MITRE ATT&CK 攻擊手法，兩者之間存在著很深的語意差距（Semantic gap）。
* 訓練樣本難以取得。模型需要需入大量標記好的攻擊資料才有機會學習到特定的攻擊行為模式
* 訓練樣本要夠多樣化。因為一個攻擊手法背後可以由不同的攻擊方法（Procedure）達成，訓練樣本的多樣性會影響到偵測模型的能力。

這也導致要在文獻中找到​​基於深度學習的 APT 偵測系統是比較少的，幾個知名且有採用深度學習的偵測模型（2020 Unicorn[11] / 2021 Atlas[12] / 2022 ShadeWatcher[13] / 2022 ProGrapher[14]）皆是在做異常偵測（Anomaly Detection），也就是只分辨 Attack / Benign。

由此可看出，使用深度學習模型進行 APT 偵測更是一項非常複雜且龐大的任務，其中涵蓋多個關鍵步驟，如有效且多樣化的攻擊樣本蒐集、轉換低階系統日誌為溯源圖（系統資源交互圖）、溯源圖的精簡化（Graph Reduction、去除冗餘資訊）、特徵工程（將原始數據轉換為適用於深度學習模型的特徵表示、文字嵌入）、深度學習模型的訓練和優化等。

最後，不論是使用基於規則或是基於深度學習的 APT 偵測方式都有其優劣，也都有機會達成不錯的偵測效果。因此在我有限的研究時間內，我實作了兩類型的偵測方式並進行比較，這項研究的成果將有助於實際了解基於規則與基於深度學習模型兩類的 APT 偵測方式，也期望這項研究能為相關團隊在未來的研究工作中提供有價值的參考和啟發。

## 【研究目標】

本論文的主要目標是透過分析系統日誌來識別惡意活動，並同時標註 MITRE ATT&CK Technique(s)。

為了實現這一目標，我們採用了兩種偵測方式。

首先，我們採用基於深度學習模型（Deep learning model）的偵測方式。深度學習模型利用神經網路架構來處理複雜的資料和模式識別。我們使用大量標記的訓練數據來訓練這個模型，使其能夠自動學習系統事件之間的相關性和惡意攻擊的行為模式。這樣的做法有助於減少對專家手動建立規則的依賴，並提高偵測規則產生和效率。

其次，我們使用開源的 Sigma 規則[15] 作為規則來源，並創建了一個基於這些規則的偵測方法，這些規則可以應用在大部分的系統日誌上。Sigma 規則是一種通用的規範語言，用於描述特定事件或惡意行為的特徵。透過這些公開規則，我們能夠快速捕捉和辨識已知的攻擊模式和技術。

綜合以上兩種偵測方式，我們能夠更全面地分析系統日誌，並及時發現可能的惡意活動。這不僅有助於提升系統的安全性和穩健性，也有助於保護重要資源和數據免於不斷演進的威脅。

## 【背景知識】

### Audit Log or System Log（未修改，可以先不用看）

（？請確認是system log or audit log?）

電腦的審核日誌（audit log）是針對一電腦作業紀錄其活動或是涉及特定活動之事件之記錄。通過記錄誰執行什麼活動以及系統如何回應等事件，再藉由檢視監視這些系統層級的事件，我們可以有效地偵測到可疑或是可能的惡意攻擊的跡象。審核日誌在不同的設備、不同的應用程序和不同的作業系統間有不同的內容紀錄與格式之差異。一個系統是否運作在正常且安全的狀態，是系統管理員以及運行於系統上的應用程式所關切的。為了能夠有效偵測系統是否有可疑或是惡意的活動正在進行，一種有效的方式是從系統審核日誌中搜尋、辨識以偵查出可識別的攻擊作業或是行為。

—---上面段落的修改版-------- （你說的和我的知識不是很吻合，尤其紀錄的內容。如果你要提Windows 系統，那請直接參考MS Windows 的 system log 的紀錄內容！？ 請給參考文獻）

電腦的系統活動日誌（System Log）紀錄了作業系統的運行狀況以及各種應用程式和服務執行情況，其中包含有關用戶登入和登出、檔案和登錄鍵訪問、網路連接、系統配置更改等事件的資訊。通過分析這些日誌，我們可以捕捉到一個程序（Process）在檔案（File）、登錄鍵（Registry）、網路（Network）等各方面資源的執行情況，通過檢查這些日誌，我們可以獲得關於程序在背後執行的行為的詳細資訊，並深入了解進程與系統之間的交互作用。一個系統是否運作在正常且安全的狀態，是系統管理員以及運行於系統上的應用程式所關切的。為了能夠有效偵測系統是否有可疑或是惡意的活動正在進行，一種有效的方式是從系統活動日誌搜尋、辨識以偵查出可識別的攻擊作業或是行為。

### Sigma Rule

Sigma 是一種開源規則格式和語言，用於描述安全事件和日誌，其規則格式非常靈活，易於編寫[圖1]，適用於任何類型的日誌文件。 Sigma 的主要目的是提供一種結構化形式，讓研究人員或分析人員可以用該格式描述他們曾經開發的檢測方法並使其可與他人共享。由於 Sigma 社群十分活躍，規則的更新頻率極高，本研究使用的規則更新至 2023/07/08，我們共下載了 1428 個針對 Windows 系統的偵測規則[[1]](#footnote-0)，這些規則可偵測出 236 個 MITRE ATT&CK Technique（攻擊手法）。

## 【研究過程－步驟概覽】

本研究進行步驟共有三個階段：前置作業、DL偵測模型建立、Sigma 規則偵測應用。

我們的**前置作業是生成系統日誌資料集 (data sets)**，其中包含對每一事件標記 Benign 或是 Attack Pattern 的標籤。**第二階段是DL偵測模型建立**，該階段分三步驟：第一步，由於前階段生成的系統日誌資料集中攻擊資料的數量有限，不足以充分訓練 DL 模型。因此，我們使用了一種置換實體屬性內的 IOC 的方法，來增加具備 Attack Pattern 標籤的事件資料。這樣可以有效提高訓練資料的樣本數，增加模型的訓練效果。第二步，為了將日誌中的文字轉換為模型可理解的格式，我們進行了文字嵌入（embedding）的處理。這個步驟將文字轉換成數值向量，以便於深度學習模型進行進一步的處理和分析。第三步，我們選用了三個不同的深度學習模型，包括多層感知機（MLP）、循環神經網路（RNN）和圖像化神經網路（GNN），來進行 Attack Pattern 標籤的偵測。**第三階段則是 Sigma 規則偵測應用**，Sigma 規則是由專家撰寫並供人閱讀的，因此我們需要一個轉換器來將這些規則轉化為電腦可識別的正則表達式語法。為了加快偵測速度，我們採用了 Multi-process 的方式進行處理。

### 前置作業. 生成對每一事件標記 Benign 或是 Attack Pattern 標籤的系統日誌資料集

我們使用 MITRE ATT&CK 維護的紅隊演練工具 Caldera[16]進行實戰攻擊，該工具提供了各個攻擊手法（Technique）的攻擊劇本（Ability[17]），一個攻擊手法底下包括多個攻擊劇本，一個攻擊劇本包括了要執行的指令（Command）、該指令可以運行的平台（Platform）、該指令的執行者（Executor）以及要包含的腳本（Payloads）。如圖2 所示，T1007 System Service Discovery 這個攻擊手法，Caldera 提供了 3 種攻擊劇本去實現，圖3 展示第三個攻擊劇本 System Service Discovery - net.ext 的內容。

此次模擬攻擊，我們執行了 167 個攻擊劇本，其中包含 12 Tactics 底下的 78 個 Techniques。在整個的攻擊過程中。被攻擊的機器數量是一台，使用 Windows 10 作業系統。總錄製過程共 24 小時，期間，被攻擊的機器仍會執行一般例如開檔案、上網等動作。

細究圖2 T1007 System Service Discovery 這個攻擊手對應的 3 種劇本所使用的指令：

* 第一個劇本—「 Discover system services」 執行 *Get-Service*
* 第二個劇本—「System Service Discovery 」執行 *tasklist.exe && sc query && sc query state= all*
* 第三個劇本—「 System Service Discovery - net.ext 」執行 *net.exe start >> C:\Windows\Temp\service-list.txt*

可以觀察出，儘管這三個劇本使用不同的指令，但它們的目的都是為了獲取系統資訊。這突顯出一個重要觀點，即一個攻擊手法可以由不同的多種攻擊模式(Attack Pattern)達成 ，即不同的方法和指令來達成相同的目的。這也意味著在建立攻擊手法辨識的模型，要能對不同的攻擊模式識別出乃是使用同一個攻擊手法，以更有效地防範和應對攻擊。

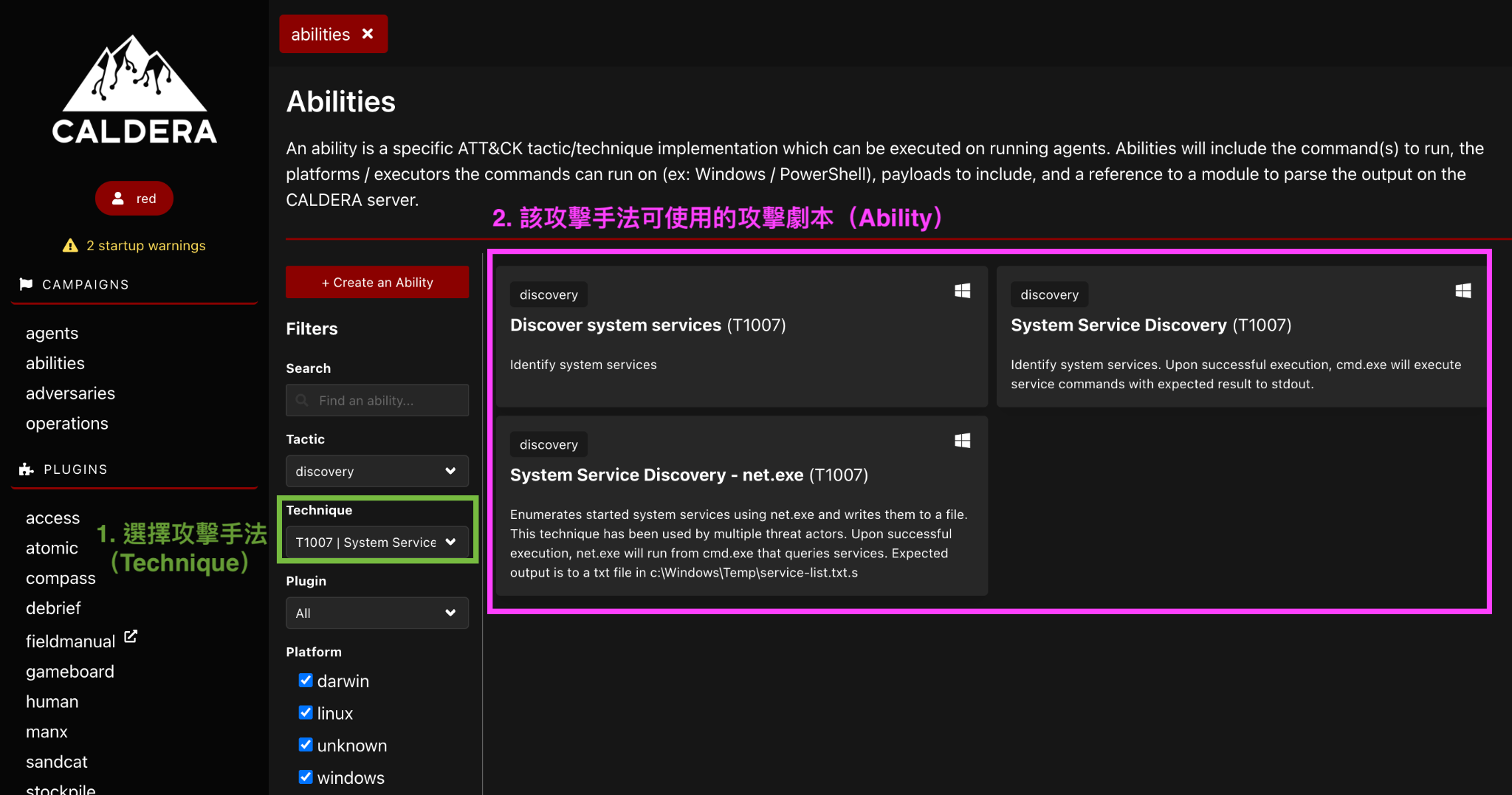


圖2. Caldera 實際操作的頁面，以 T1007 System Service Discovery 該攻擊手法為例



圖3. Caldera 攻擊劇本的頁面，以 System Service Discovery - net.ext 該攻擊劇本為例

為了蒐集被攻擊系統的系統日誌，在執行攻擊演練的同時，我們利用 Process Monitor[18] 錄製被攻擊系統的系統日誌。Process Monitor 是一個由 Microsoft 開發的強大系統監控工具，旨在幫助用戶即時監控和紀錄 Windows 作業系統中的各種系統活動和程序（process）。本研究透過此工具蒐集檔案系統、登錄鍵、網路等活動紀錄，整理出系統日誌作為本研究的研究資料集。

一個系統日誌由多個系統事件（System Event）組成，如圖4所示。一個系統事件為圖中的一行。每一個系統事件實際上由 27 個欄位（詳見圖5）組成，圖4僅呈現了我們選擇的 9 個欄位：

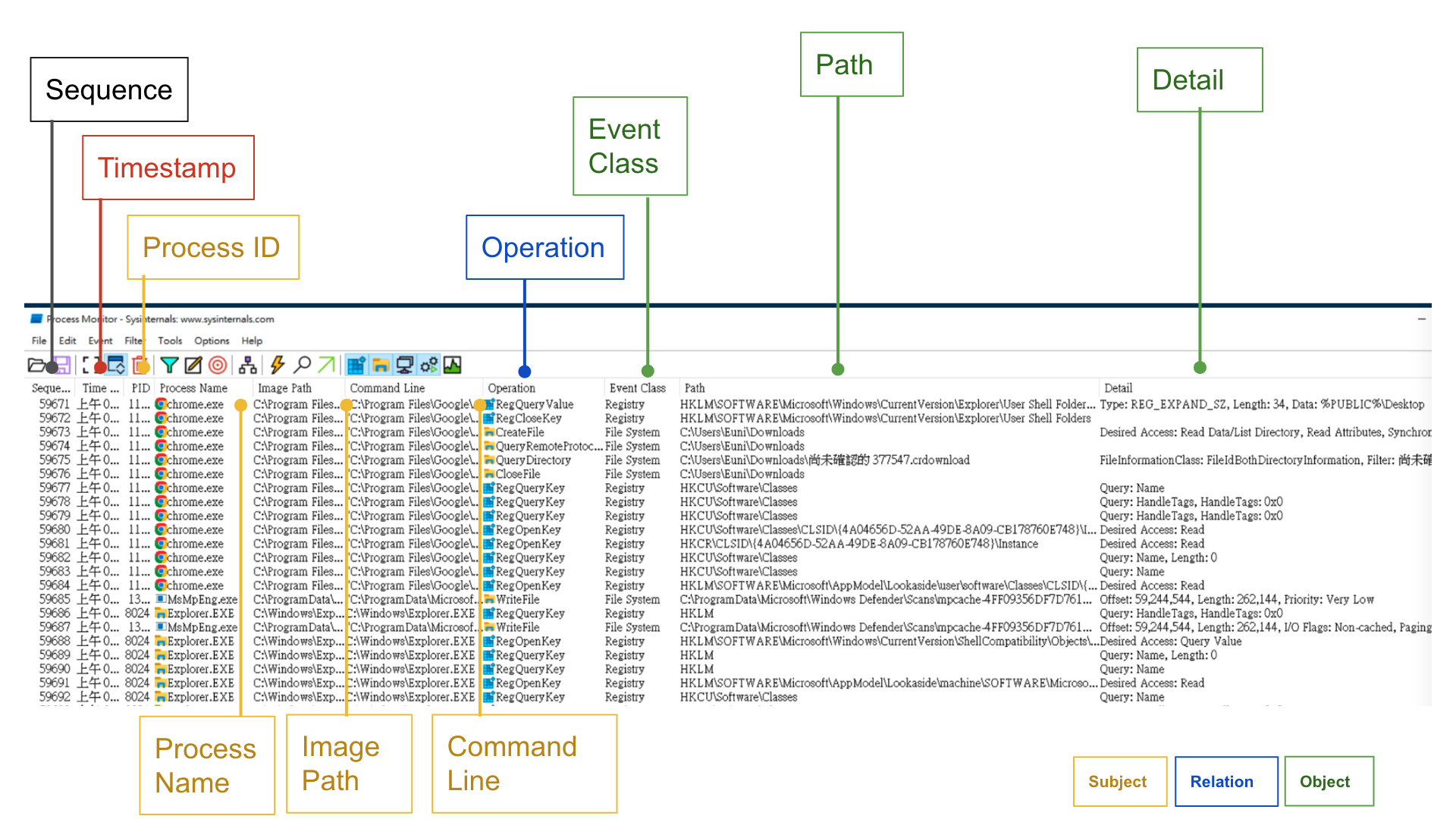
1. Time of Day（時間）：操作發生的時間。
2. Process ID（程序 ID）：執行操作的程序的 ID（PID）。
3. Process Name（程序名稱）：事件發生所涉及的進程的名稱。
4. Image Path（映像路徑）：運行在進程中的映像（執行檔）的完整路徑。
5. Command Line（命令行）：用於啟動進程的命令行。
6. Operation（操作）：具體的事件操作。
7. Event Class（事件類別）：事件的類別（檔案、註冊表、進程等）。
8. Path（路徑）：事件所參考的資源的路徑。
9. Detail（詳細資訊）：特定事件的附加資訊。

觀察了所有欄位後，我們發現這九個欄位可以幫助我們進一步將一個系統事件整理成「Subject entity執行了某一個 Operation ，作用在 Object entity」的格式，這有助於更清楚地描述和理解事件的內容和系統資源間的關聯。

具體而言，我們可以通過以下方式來整理和解釋系統事件：

* Subject entity：事件中執行操作的主體，即觸發該事件的程序或程式。
* Operation：表示該事件所執行的具體操作，例如讀取、寫入、創建等。
* Object entity：表示該事件所作用的受體，即受到該操作影響的資源，可能是檔案、註冊表、網路連接等。

透過這種整理，我們能夠更清楚地看到每個事件的主要參與者、執行的操作和作用的對象，從而更深入地理解系統的運作和系統資源之間的關係，為後面階段的事件追蹤與分析提供了更清晰的框架和基礎。

圖4. 系統活動日誌在 Process Monitor 所顯示的畫面

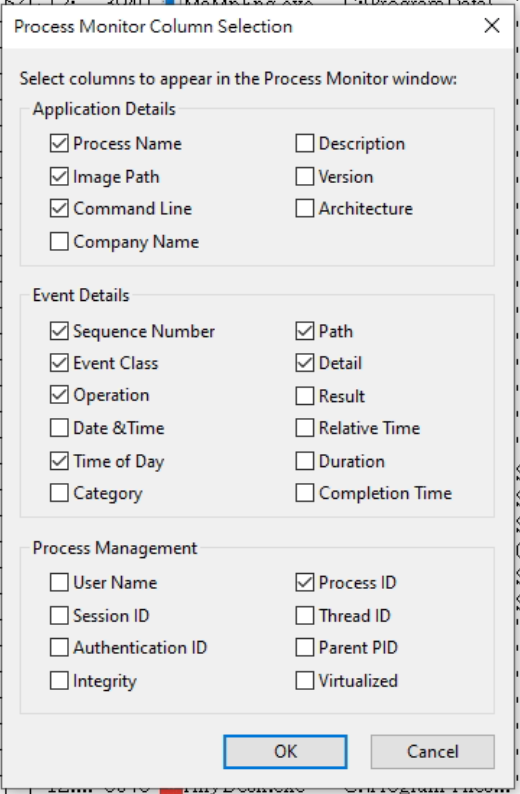
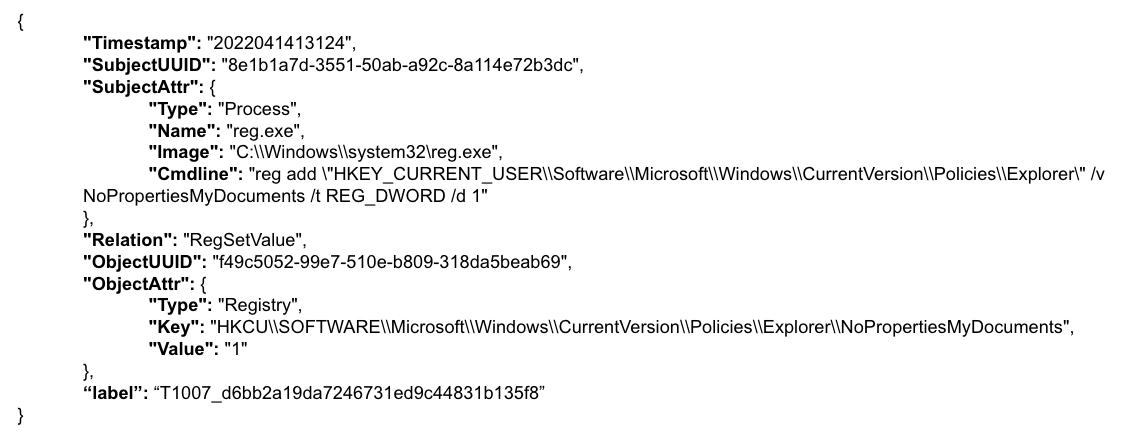


圖5 Process Monitor 的所有欄位，打勾的為選擇的欄位

然而，Process Monitor 所紀錄的系統事件並不會明確標註來自 Caldera 哪一個攻擊劇本的產生，為了解決這個問題，我們參考執行 Caldera 後生成的攻擊報告[19]。在報告中，記錄了各個攻擊劇本產生的 Process ID 以及執行者的名稱。藉由這些資訊，我們可以將攻擊劇本的執行者與 Process Monitor 紀錄的 Process ID 和 Process Name 欄位對應起來，進而確定哪些系統日誌事件是由 Caldera 的特定攻擊劇本所產生的。這樣我們就能對系統日誌裡的事件一一標記其為Benign 或是Attack Pattern 標籤。

總的來說，我們將 Process Monitor 錄製到的系統事件中選取9個重要欄位，將其整理整 Subject entity / Relation(Operation) / Object Entity 的形式，並將其與 Caldera 的劇本 ID 對應，整理成了 JSON 格式，如 圖6 所示。表1 列出整理後的事件屬性與對應回原本 Process Monitor 的欄位。

圖6. 整理好的系統事件格式，此圖為一個系統事件包含的所有資料

| **系統事件屬性** | **來源** | **來源欄位** | **定義** |
| --- | --- | --- | --- |
| Timestamp | Process Monitor | Time of Day | 該事件發生的時間戳。 |
| Subject Attributes | Process Monitor | Process ID、  Process Name、  Image Path、  Command Line | Subject entity的屬性 |
| Subject UUID | Process Monitor | Process ID、  Process Name、  Image Path、  Command Line | Subject entity的唯一識別碼 |
| Object Attributes | Process Monitor | Event Class、  Path、  Detail | Object entity的屬性（根據不同的系統資源種類會有不同屬性） |
| Object UUID | Process Monitor | Event Class、  Path、  Detail、 | Object entity的唯一識別碼 |
| Relation | Process Monitor | Operation | Subject entity 對 Object entity的操作 |
| Label | Caldera | 劇本ID | 透過對應 PID 的方式填上標籤 |

表1 整理後的系統事件屬性對應表

在這些事件中，Subject entity和 Object entity 都是屬於系統資源的實體。為了辨識和區分這些資源，我們新增Subject entity 和 Object entity的唯一識別碼（UUID）。這使得我們在後續分析多個系統事件時，透過觀察 UUID，可以判斷不同事件之間的 Subject entity和 Object entity 是否有關聯。

例如，圖6 範例事件中的 Object entity’s UUID 是”f49c5052-99e7-510e-b809-318da5beab69”，如果在其他系統事件中也出現一樣的，那就表示不只一個系統資源在對 "HKCU\SOFTWARE...MyDocuments" 這個登錄鍵進行操作。

Process Monitor 錄製到的系統資源類別有 4 種：程序（Process）、檔案（File）、登錄檔（Registry）以及網路（Network）。而 UUID 的設計方式是將各資源類型的屬性（見表2），串接後再經過一個 Hash Function（雜湊函數）。例如，若資源類型為 Process，則將其程序 ID（Process ID）、程序名稱（Process Name）、映像路徑（Image Path）以及命令行（Command Line）串接後進行 Hash；而若資源類型為登錄檔 ，則將登錄鍵（Registry Key）以及登錄值（Registry Value）串接後進行 Hash。這種 UUID 的設計方式可以讓我們確保一個 UUID 僅代表一個系統資源，同個 UUID 的系統資源屬性也都會相同。

| **程序 Process** | |
| --- | --- |
| Process ID | 進程編號 |
| Process Name | 進程名稱，如：  \*.exe 的執行檔  \*.dll 的動態函式庫 |
| Image Path | 進程名稱的完整路徑 |
| Command Line | 用於啟動進程的命令行 |
| **檔案 File** | |
| File Name | 檔案完整路徑 |
| **登錄檔 Registry** | |
| Registry Key | 登錄鍵完整路徑 |
| Registry Value | 登錄值 |
| **網路 Network** | |
| Source Address | 來源位址  Domain 或 IP + (Port) |
| Destination Address | 目標位址  Domain 或 IP + (Port) |

表2 整理後的各類型系統資源對應到的屬性

在這個階段，我們共蒐集了 17,346,525 個系統事件，其中有 17,328,191 個事件被標記為正常（Benign），有 18,334 個事件被標記為攻擊劇本ID。

然而，我們觀察到蒐集來的系統事件中存在許多重複的情況（見圖7）。這是因為 Process Monitor 捕捉事件的頻率非常快，導致大量重複的系統事件，這在後續的系統資源互動分析中帶來了困難。因此，我們對重複的資料進行了基本的精簡。在不考慮時間戳（Timestamp）的情況下，僅保留 Subject UUID、Subject Attributes、Object UUID、Object Attributes、Relation 和 Label 六個屬性。因此，系統事件數量減少至 4,932,604 筆，其中有 4,916,024 個被標記為正常（Benign），而 16,580 個被標記有攻擊劇本 ID 標籤。

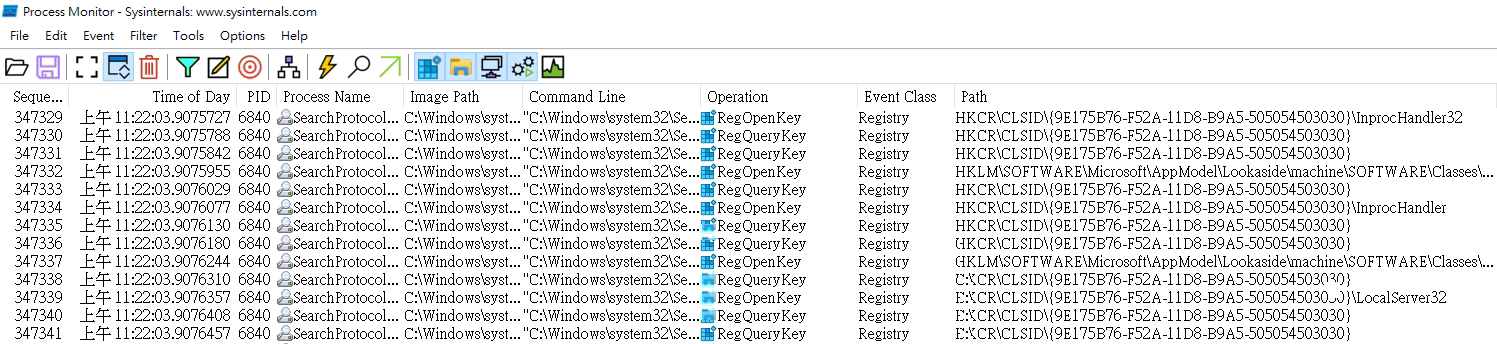
**

圖7. 含時間戳記的系統事件

在後續的研究中，我們可能會使用含有時間戳的系統事件或不含時間戳的系統事件進行分析。為了說明清楚，我們將移除時間戳的系統事件稱為「三元組」（Triplet）。

「三元組」(triplet) 的概念源於知識圖譜（Knowledge Graph），最早由 Google 於2012年提出[20]，其目的在於將原本的文字資訊進行結構化，透過抽取、存儲和關聯原始知識，形成多個「實體-關係-實體」的三元組。這樣的設計使得知識可以以圖的架構進行存取和關聯。典型的例子是 DBpedia[21]（見圖8），它是一個大規模的多語言百科知識圖譜，可視為維基百科的結構化版本。

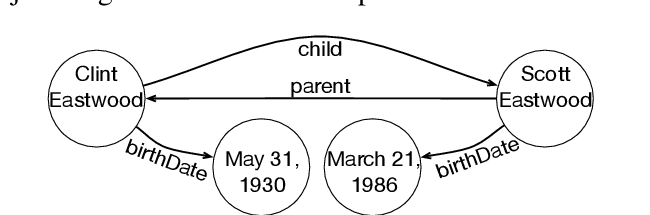
**

圖8. 知識圖譜範例（four triples from DBpedia.）

我們前面提到，在一系統事件中，我們所關注的是 subject entity, object entity and their relationship in terms of the operation that the subject entity performs on the object entity. 因此，在我們的案例中，我們將各個系統資源視為實體，系統動作（Operation）視為關係，這即是一個三元的關係表示，因此我們可以直接將移除時間戳的系統事件稱為三元組 。

根據圖7，在移除事件時間戳後，應該會出現許多重複的三元組，我們可以直接將這些重複三元組刪掉，讓冗餘的資訊消失，該動作並不會影響將三元組轉換成知識圖的型態。表3 呈現了系統事件與三元組的比較。圖9 則呈現圖6 事件轉換而成的三元組。在後面的研究當中，我們會將多個三元組組合成圖，該圖在本論文稱之為系統資源交互圖。圖的 Schema 請見圖10。

| 名稱 | 包含屬性 | 意義 | 數量 |
| --- | --- | --- | --- |
| 系統事件  Event | 1. src\_uuid, src\_attributes  2. dst\_uuid, dst\_attributes  3. relation  4. timestamp | 該組系統資源互動過程的紀錄檔（含時間戳） | 17,346,525 |
| 三元組  Triplet | 1. src\_uuid, src\_attributes  2. dst\_uuid, dst\_attributes  3. relation | 一組系統資源的作用關係＝移除時間戳的系統事件 | 4,932,604 |

表3 系統事件與三元組比對

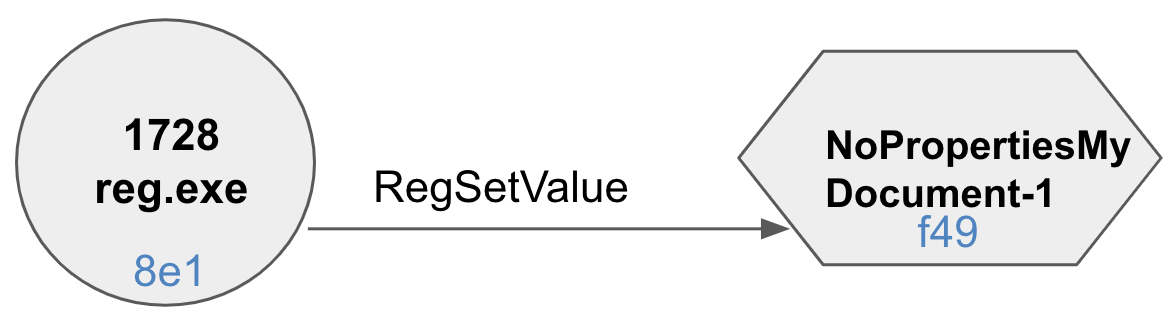
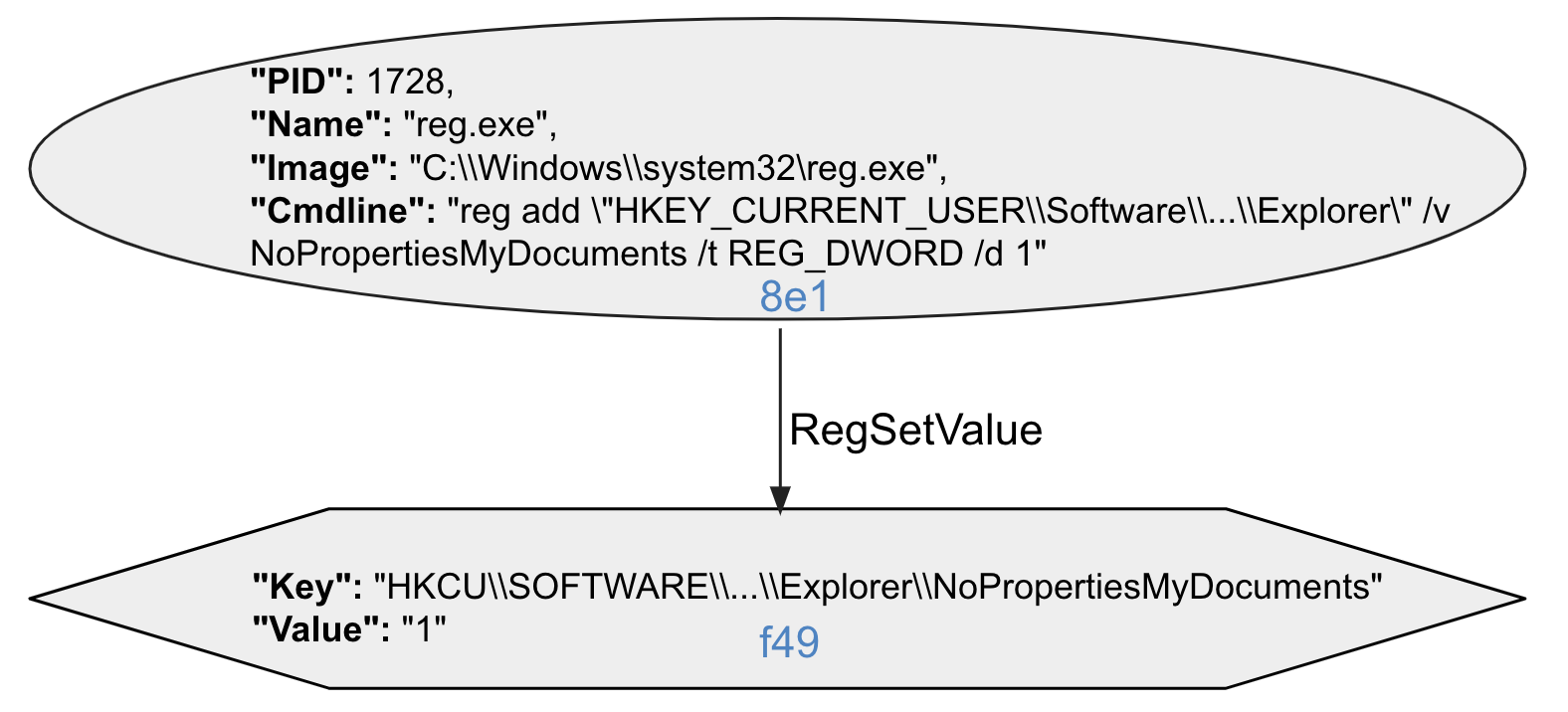
**

圖9. 圖6轉換而成的三元組（左圖是實際上的樣子，右圖為簡化版）。藍字代表 UUID。

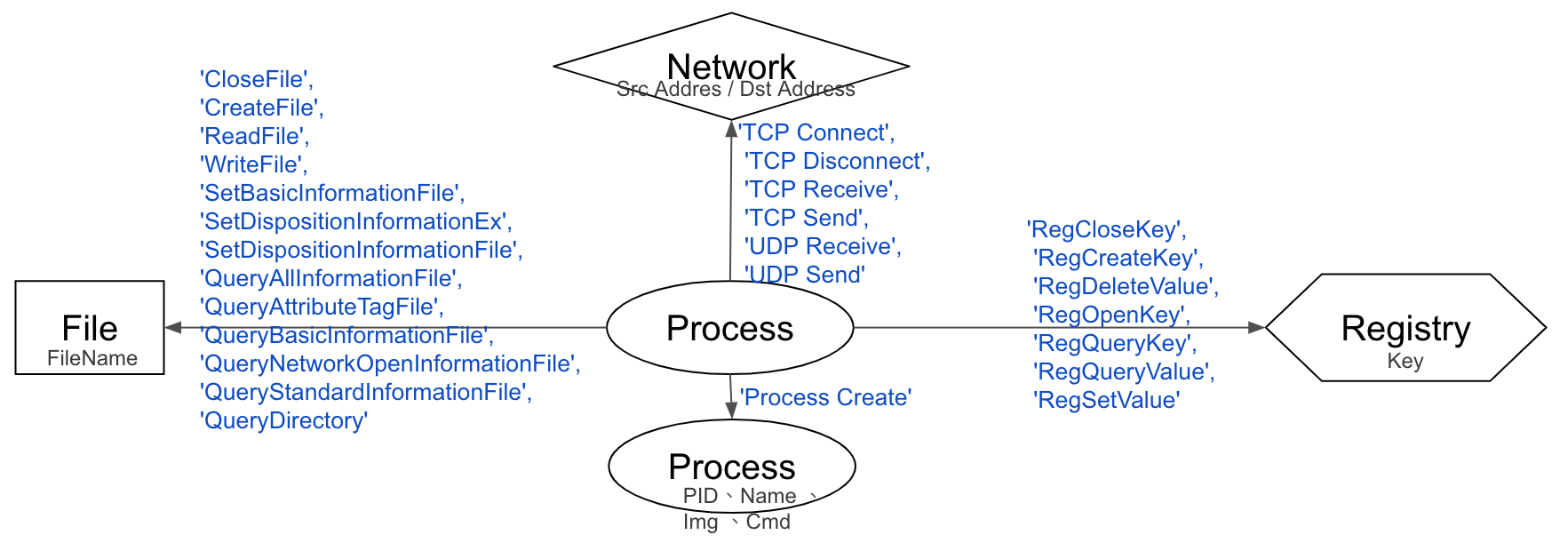
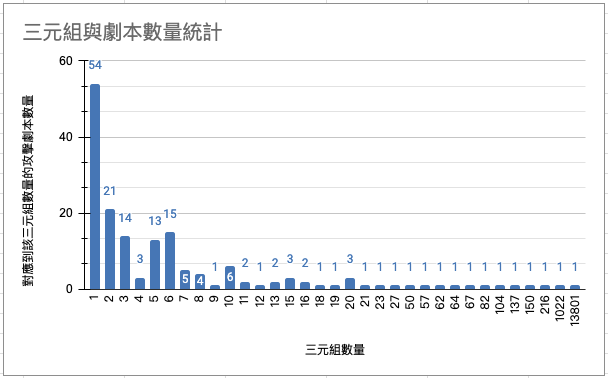
**

圖10. 本論文系統資源交互圖的 Schema。節點的形狀代表不同的實體資源（圓圈： Process 、六角形： Registry、方形：File、菱形：Network），不同的節點有不同的屬性。邊則代表 Process Monitor 中的 Operation。

### DL 偵測模型：增加具備 Attack Pattern 標籤的系統日誌事件資料集

在研究中，我們針對 167 個劇本進行了資源互動事件的統計分析，並以三元組作為代表來了解這些劇本所產生的資源互動情況。根據統計結果（如圖11所示），我們觀察到以下現象：

首先，大部分的劇本都僅對應到 6 個以內的三元組，尤其是有將近三分之一的劇本僅對應到 1 個三元組，這表示這些劇本所造成的資源互動非常有限，可能僅包含單一的作用事件。附錄整理了各個攻擊劇本對應到不同三元組數量的案例。

圖11. Caldera 攻擊劇本對應到的三元組數量統計（有 54 個劇本僅對應到 1 個三元組，21 個劇本對應到 2 個三元組，以此類推。）

然而，不論一個劇本對應到幾個三元組，一個攻擊劇本在一台機器上只會產生一份系統日誌；，167 個攻擊劇本會產生 167 份符合該劇本攻擊模式的系統日誌。由於下階段模型的目標是要建立針對各個攻擊模式的分類器，也就是一個有 167 個類別的分類問題，目前每一個類別都只有一筆資料，該資料量在訓練DP模型時嚴重不足。為了新增各個類別底下的資料筆數，我們採用置換三元組中實體的 IOC、Process ID(PID)、UUID 的方式增加資料量。 （？老實說，我不覺得這個方法是好方法，因為攻擊行為的本質都沒變，能學習到的知識是一樣的。）

人工觀察三元組中的各類別實體的屬性（見表2），我們發現在 command line、file name、src/dst address 中很容易出現可以置換的屬性，如圖12 所示，T1016\_Remote\_host\_ping 這個劇本利用 Ping 檢查遠端主機是否可訪問，其主機位置即是可置換的 IOC。為了要讓置換後的資料更符合攻擊情況， 我們置換的 IOC 列表是由其他實驗利用 Cuckoo Sandbox[22] 在錄製 17924 隻惡意程式得到的分析報告[23]中「Signature」的欄位取得（見圖14），然而，以現有的分析報告並無法取得所有類別的 IOC，因此，部分 IOC 類別是使用 Faker[24] 這個套件來生成假的 IOC。置換 IOC 的類別整理至表4。

屬於 Process 類別的實體有 PID 這個屬性，在實際運行的電腦中，作業系統會為每個程序分配一個隨機的 PID，因此我們也會為不同程序給上一個隨機非負整數數當作 PID（見圖12/圖13），此舉除了模擬真實情況，更重要的用途是讓一些無法有置換 IOC 的實體可以得到一個不一樣的 UUID，使得新增的資料可以在圖中產生新的三元組（見圖13）。此外，因為新增的資料必須在系統交互圖上產生新的資料，因此我們在每個 UUID 的前面都加上 *fake{i-1}-* 來代表它是第 i 個新增的資料。

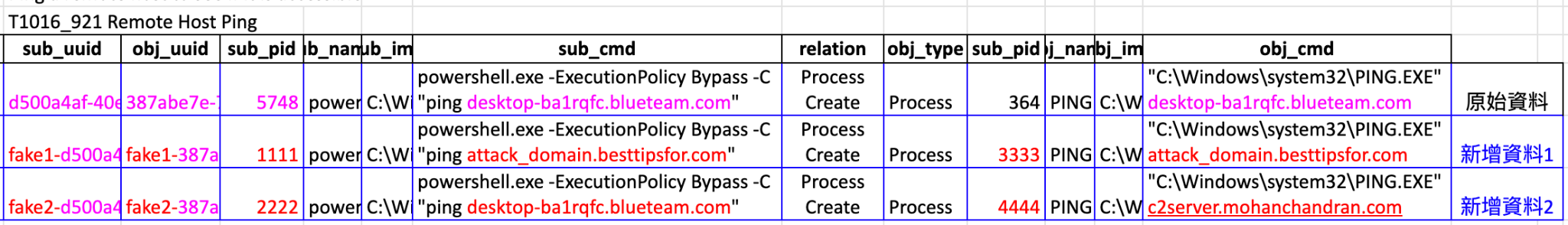


圖12(A). 劇本 T1016\_Remote\_host\_ping 利用 Ping 檢查遠端主機是否可訪問。該劇本由一個三元組組成，內容是一個主體 powershell process 建立一個受體 ping process，兩者的 command line 都包含了 host IOC。粉色字體為原始 IOC，紅色字體為置換後的 IOC。

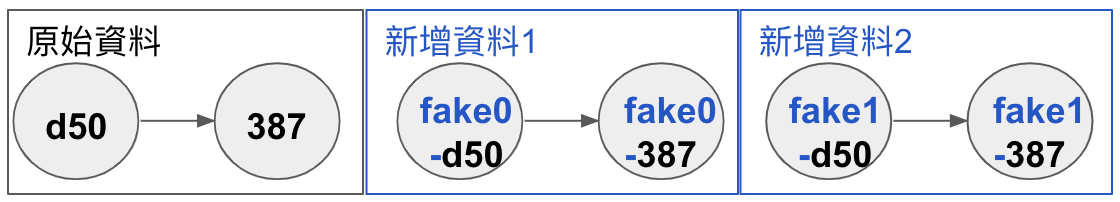


圖12(B). 以系統交互圖的方式呈現圖12(A)，圈內文字為 UUID。

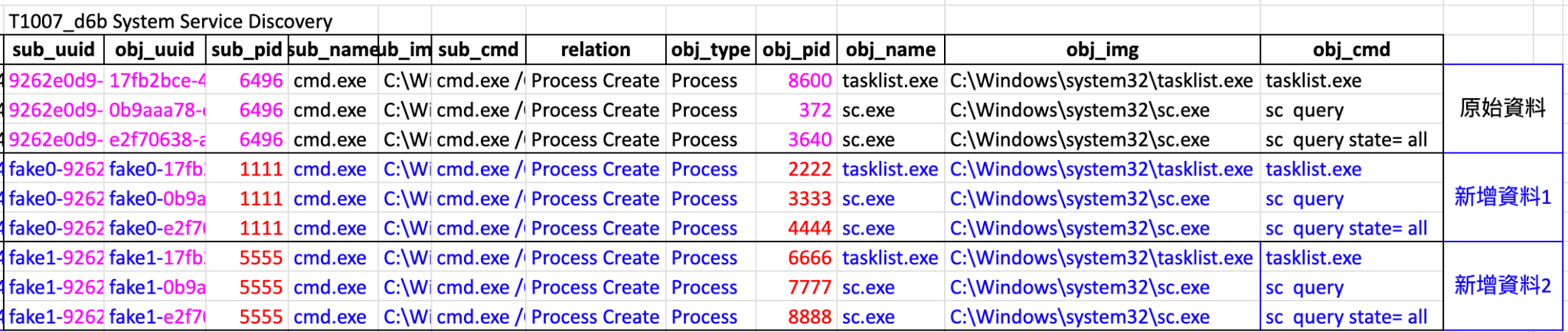


圖13(A). 劇本 T1007\_System\_Service\_Discovery 對應到 3 個三元組及 4 個實體，這 4 個實體皆沒有合適的 IOC 可以置換，但透過置換 PID，仍可以達到 UUID 的設計原則，即一個 UUID 對應到一個實體且其所有屬性的串接字串是唯一。

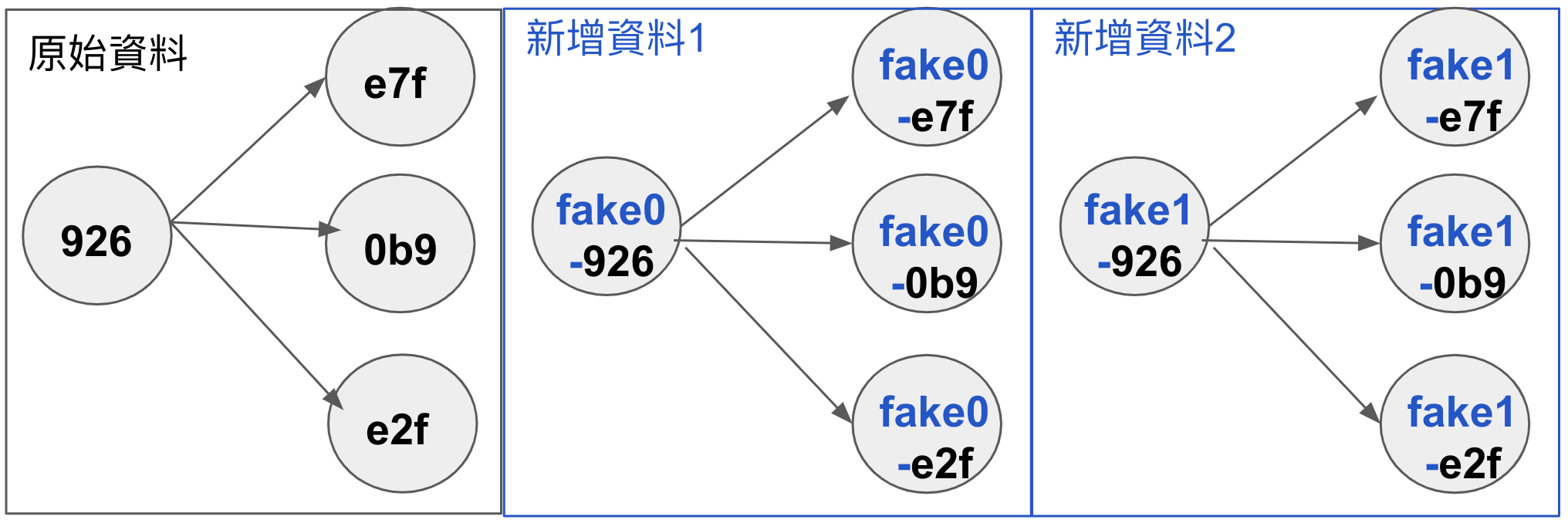


圖13(B). 以系統交互圖的方式呈現圖13(A)，圈內文字為 UUID。

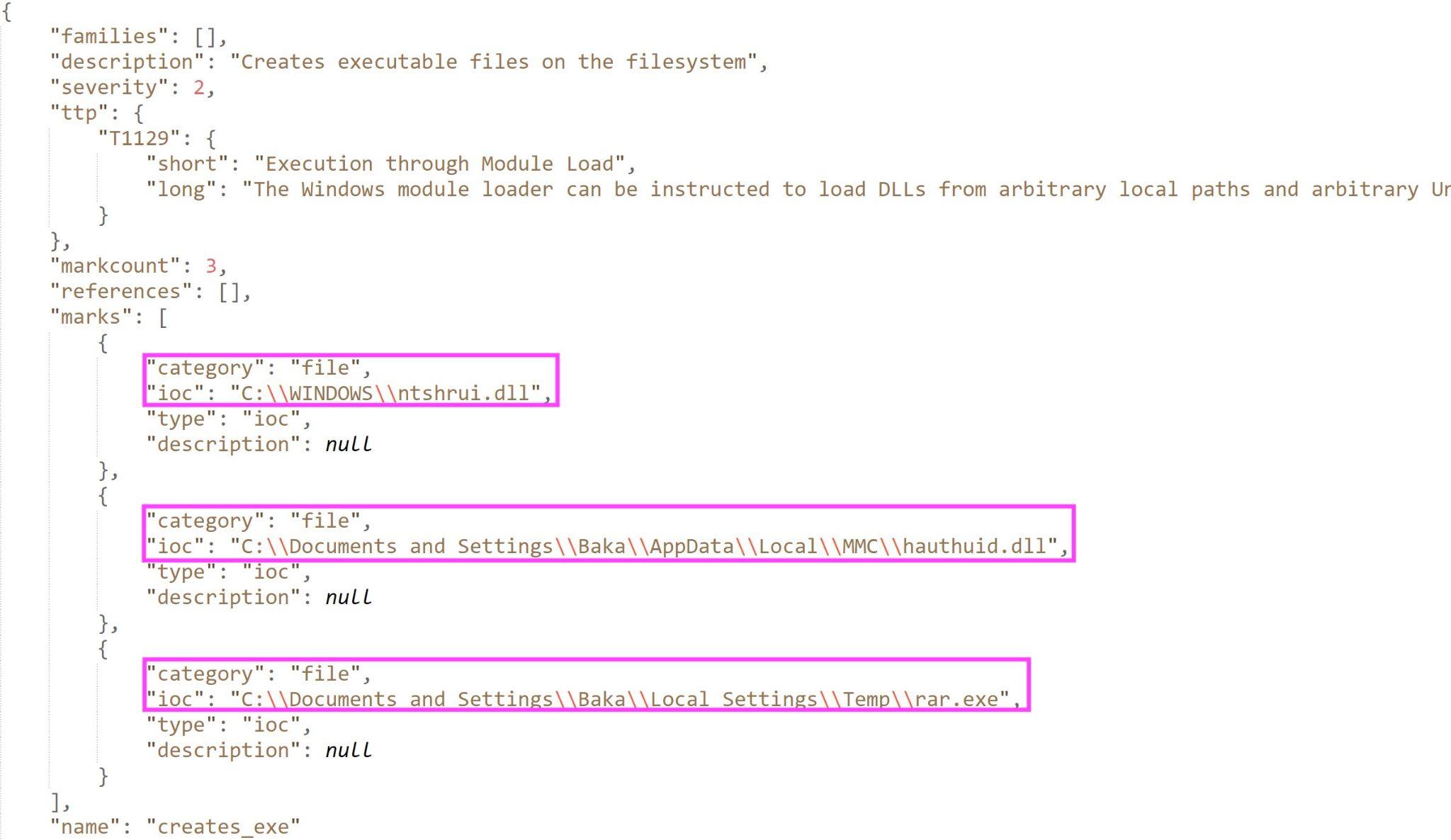


圖14. Cuckoo Sandbox 執行惡意程式得到的 JSON 分析報告（欄位：signatures）

| 類別 | 來源 | IOC 數量 | IOC 列表（例子） |
| --- | --- | --- | --- |
| \*.exe | Cuckoo Sandbox | 1688 | * reader\_sl.exe * regsys.exe * searchprotocolhost.exe * setup.exe * ef2f8d90ebae12c015ffea41252d5e25055b16c6.exe * … |
| service  name | Cuckoo Sandbox | 328 | * GoogleChromeElevationServiceSys * FastUserSwitchingCompatibility * PCAudit * PNRPAutoRegSys * PNRPsvcSys * … |
| command  line | Cuckoo Sandbox | 9106 | * C:\Documents and Settings\Baka\Local Settings\Temp\tmp.docx * C:\Documents and Settings\All Users\Application Data\FlashFXP\3\History.dat * C:\Users\Baka\AppData\Local\Temp\1142DCC02B9EF34DCA2F28C22613A0489A653EB0AEAFE1370CA4C00200D479E0.xlsx * … |
| host | Faker | 無限 | * lt-04.sanchez.biz * srv-03.garcia-ross.net * lt-93.sanders.org * srv-47.watson.com * db-53.munoz-perkins.com * … |
| domain | Faker | 無限 | * stevens.com * buchanan.biz * roberts.org * luna.com * … |
| \*.dll | Faker | 無限 | * D:/project/one/enter/popular/occursing.dll * C:/if/current/rock/born/floortend.dll * C:/just/together/summer/recognize/helike.dll * C:/wait/lose/allow/house/beatduring.dll * … |

表4. 置換 IOC 列表整理

經過上述的處理，我們解決了原始資料集中 167 個攻擊劇本都僅各自有一份系統日誌的議題，並將 167 個攻擊劇本都各自都生成出 1000 份的資料。

### **DL 偵測模型：文字嵌入（embedding）**

文字嵌入（embedding）是在將資訊輸入模型前的一個關鍵步驟，它將文本或符號型數據轉換為數值向量的形式，以便於深度學習模型的處理與分析。在我們的研究中，我們採取了兩種不同的方法來進行文字嵌入，以更好地代表三元組的含義和主題語義。

#### **文字嵌入 — TransX**

首先，我們考慮了三元組的基本結構，即「subject entity 執行了某一個 operation，作用在 object entity」。這種方法旨在將三元組的語義關聯和結構轉化為數值向量，以便於深度學習模型的理解和學習。為此，我們採用了TransE、TransH、TransR 三個模型[25][26][27] 來實現。

具體來說，TransE 模型將每個實體和關係表示為一個固定維度的向量。對於一個給定的三元組 (h, r, t)，其中 h 表示頭實體（head entity）、r 表示關係（relation）、t 表示尾實體（tail entity），TransE 模型的目標是使得 h + r 大約等於 t。這種約束（constraint）意指，關係 r 可以被視為頭實體 h 轉移到尾實體 t 的向量表示。換句話說，TransE 模型假設一個有效的三元組應該在向量空間中滿足這樣的線性關係（如圖14所示）。

訓練 TransE 模型的方法是通過最小化所有已知三元組和負例三元組[[2]](#footnote-1)之間的距離（例如，使用 L1 或 L2 距離），使得已知三元組滿足 h + r ≈ t，同時負例三元組則不滿足這樣的關係約束。這樣的訓練過程能夠使得實體和關係的向量表示逐漸適應到知識圖譜中的關係結構。

公式1 表示了 TransE 模型的得分函數（Scoring Function），此公式衡量知識圖譜中一個三元組（h, r, t）是否符合模型所學習到的約束關係。這個得分函數的得分越小越好，表示頭實體加上關係的向量應該越近似於尾實體的向量。公式2 表示了TransE 模型的損失函數（Loss Function），這個損失函數的目標是最小化正確三元組的得分和負例三元組的得分之間的差距。

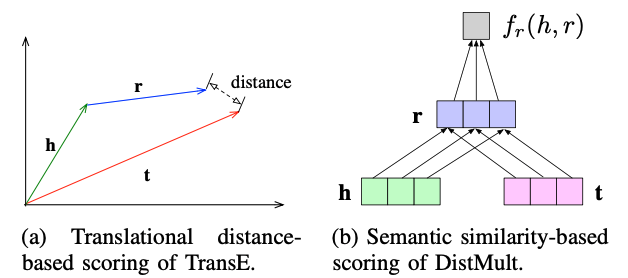
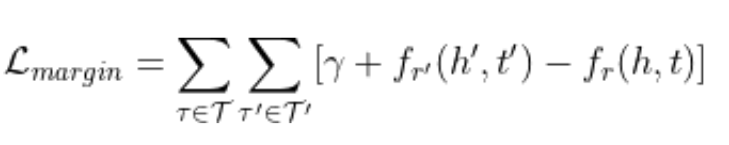


圖14. TransE 模型假設一個有效的三元組應該在向量空間中滿足 h + r ≈ t 的線性關係



公式1. TransE 模型的得分函數

公式2. TransE 模型的損失函數是 margin-based，*T* 是已知的正確的三元組集合，*T’* 是負例三元組三元組集合， 是預先定義的邊界值（margin），這個損失函數的目標是確保已知的三元組滿足得分的約束關係，同時將負例三元組的得分提高到比已知的三元組得分加上一個預先定義的邊界（margin）值以上。

儘管 TransE 模型在訓練過程中比較簡單，且易於實現，但它有一些限制。由於 TransE 將實體和關係映射到相同向量空間中，可能會忽略到一些細微的語義差異，尤其是在處理一對多或多對一的關係時。為了克服這個問題，後來提出了一些改進的模型，如 TransH 和 TransR。TransH 改將實體和關係映射到多個超平面中， 它通過引入超平面來表示不同關係的不同語義，從而更好地捕捉實體和關係之間的語義特徵，因此 TransH模型在處理多義性[[3]](#footnote-2)和多重關係[[4]](#footnote-3)時具有一定的優勢。TransR 模型則是將實體和關係映射到不同的向量空間中。它通過引入關係矩陣來捕捉實體和關係之間的相互影響，更全面地表示三元組之間的語義關係。這使得TransR模型在處理複雜的三元組關係時表現出色，並能夠有效地捕捉事件中的上下文信息。圖15 比較三個模型的的設計概念。

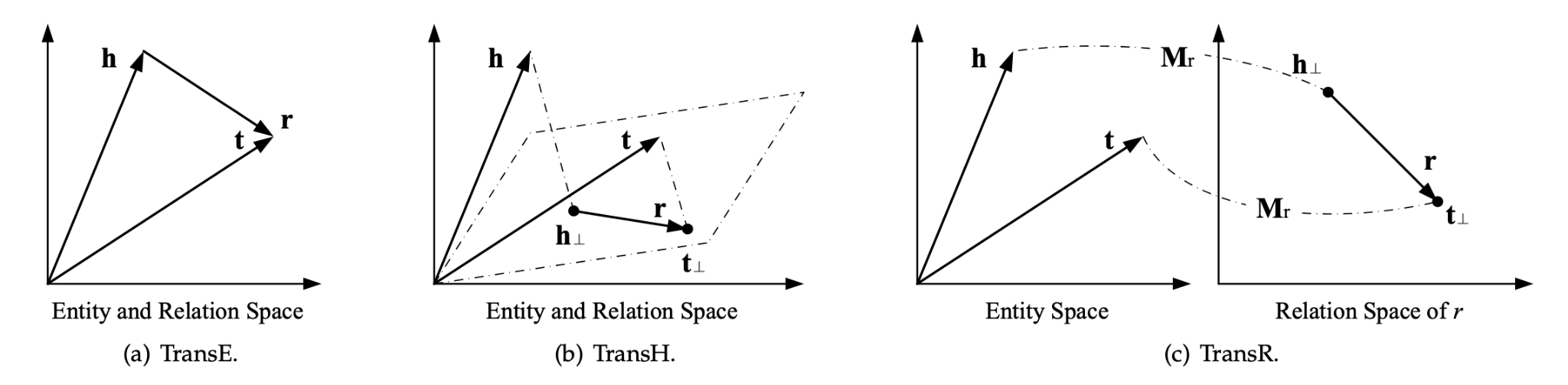


圖15. TransE、TransH、TransR 概念比較。

由於我們的資料集是一個有向異質多圖（directed heterogeneous multigraph），即包含多種實體和多種關係的多圖結構，理論上，TransR 使用關係矩陣來處理實體和關係之間的映射關係，這樣應該能更好地捕捉到複雜的實體和關係的語義關聯。然而，具體模型的效果還是會因為資料特性以及研究目標而有所不同，因此，我們決定將三個模型產生的文字嵌入都餵入下個階段的偵測模型進行實驗。這階段 TransX 文字嵌入的詳細訓練過程請見 Appendix 2。

#### **文字嵌入 — SecureBERT**

此外，除了上述考量三元組的結構語義，我們考慮了第二種文字嵌入的方式，即使用預訓練模型（pre-trained model）來精準掌握資安主題的描述內容之語義。在這種方法中，我們選擇了SecureBERT模型[28]，這是一種基於BERT（Bidirectional Encoder Representations from Transformers）[29]的預訓練模型，專門針對資訊安全領域的文本進行嵌入。SecureBERT模型在大量資安相關文本的訓練下，已經具備了豐富的語義理解能力，能夠將資安主題的描述內容轉換為 768 維的向量表示。這樣的嵌入方式能夠幫助我們更準確地捕捉到資安事件中的細節和特徵，並提高模型的偵測能力。

當文字嵌入（embedding）這步驟完成後，我們將獲得資料集中所有實體（Entity）和關係（Relation）的數值向量表示。這些文字嵌入向量將能更好地代表三元組的含義和主題語義，進而協助下階段深度學習模型將具有更強的識別能力和偵測效果。

### **DL 偵測模型: MLP/RNN/GAT（還來不及寫...）**

在經由embedding model將一系統日誌內的每一筆事件的 subject entity、 operation 、object entity 轉換成各自的文字嵌入（embedding）後，接下來是建立DL模型。我們考慮三個模型。

| 模型 | 挑選原因 |
| --- | --- |
| MLP (MultiLayer Perceptron) | 一個簡單的 Neural Network 模型，作為利用深度學習模型的基本款，評估其成效，並與rule-based approach 作比較。 |
| RNN (Recurrent Neural Network) | 一種 Neural Network 模型，其有納入考量日誌事件的時間順序。 |
| GAT (Graph Attention Network) | 一種 Neural Network 模型，事件實體(subject and object entities)以及operation以 graph 表示，可以表示事件間的交互關係。 |

| 模型 | 輸入(X) | 輸出(y) | 模型可以學到的東西 |
| --- | --- | --- | --- |
| Multilayer perceptron  (MLP) | One Triplet  (150, ) | Attack Pattern | Triplet 內的關係。 |
| Recurrent Neural Network(RNN) | Event Sequence  (32, 150) | Attack Patterns  (32, ) | 同一個 Sequence 內事件的關係 |
| Graph Attention Network  (GAT) | A graph | Node Classfication?  Edge Classfication?  Graph Classfication? | 鄰居節點之間的關係？ |

請將三個模型個別的設計作詳細的說明並畫出各模型的結構圖。

#### DL 偵測模型: MLP

#### DL 偵測模型: RNN

#### DL 偵測模型: GAT

### **Sigma Rule 偵測實驗**

Sigma Rule 偵測實驗主要分成三個步驟：規則蒐集、規則轉換、規則偵測。

#### **Sigma Rule：規則蒐集**

由於 Sigma 社群十分活躍，幾乎每天都會有規則更新，本研究的實驗結果皆是以 2023 年 07 月 08 號下載的規則為研究資料。

#### **Sigma Rule：規則轉換**

#### **Sigma Rule：規則偵測**

### **成效比較**

每一個模型都可以產生一個如下的表格：

| Label | TP | FP | TN | FN | Precision | Recall | F1-score |
| --- | --- | --- | --- | --- | --- | --- | --- |
| T1011\_劇本1 |  |  |  |  |  |  |  |
| T1011\_劇本2 |  |  |  |  |  |  |  |
| T1011\_劇本3 |  |  |  |  |  |  |  |
| T1022\_劇本1 |  |  |  |  |  |  |  |
| T1022\_劇本2 |  |  |  |  |  |  |  |
| T1022\_劇本3 |  |  |  |  |  |  |  |
| T1022\_劇本4 |  |  |  |  |  |  |  |
| T1033\_劇本1 |  |  |  |  |  |  |  |
| T1044\_劇本1 |  |  |  |  |  |  |  |
| T1044\_劇本2 |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |

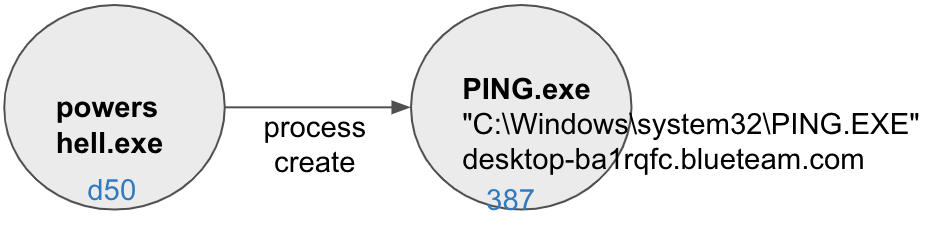
最後在綜合比較三個模型和 rule-based 產生一個如下的表格：

|  | FP | Macro-Precision | Macro-Recall | Macro-F1-score |
| --- | --- | --- | --- | --- |
| Sigma Rule |  |  |  |  |
| MLP |  |  |  |  |
| RNN |  |  |  |  |
| GAT |  |  |  |  |

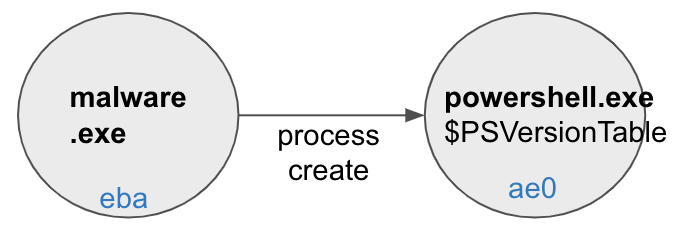
### **觀察與發現**

## 【Appendix 1 — 攻擊劇本對應到不同數量的三元組案例】

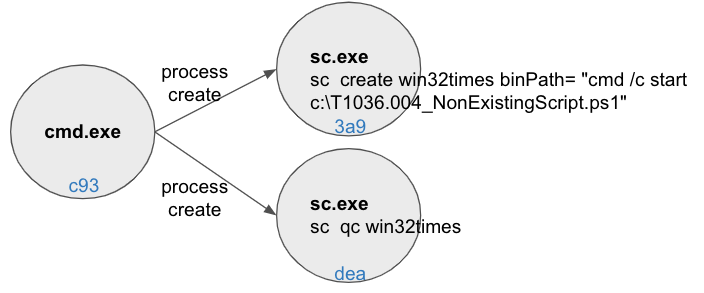
* 僅對應到「一個」三元組的劇本案例：圖A、圖B
* 對應到「兩個」三元組的劇本案例：圖C、圖D
* 對應到「三個」三元組的劇本案例：圖E、圖F
* 對應到「多個」三元組的劇本案例：圖G、圖H



圖A. 「一個」三元組的劇本案例 — T1016\_921: Remote Host Ping。該劇本目標是利用 Ping 檢查遠端主機是否可訪問，進而達到 T1016 System Network Configuration Discovery 的技巧。

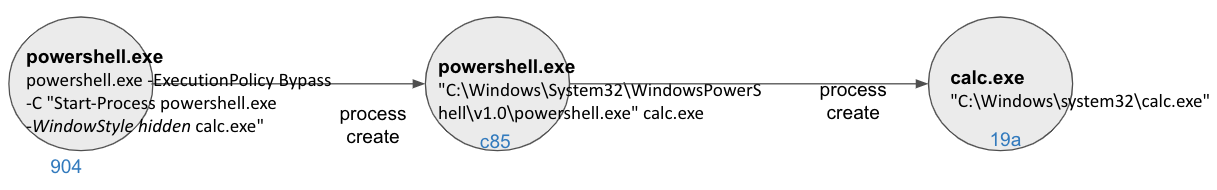


圖B. 「一個」三元組的劇本案例 — T1082\_294: PowerShell version。該劇本目標是找出 PowerShell 的版本，進而達成 T1082 System Information Discovery 這個 Technique。

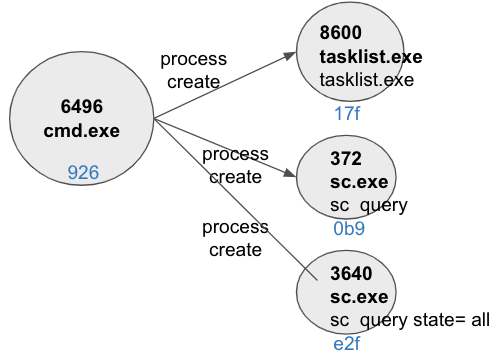


圖C. 「兩個」三元組的劇本案例 —

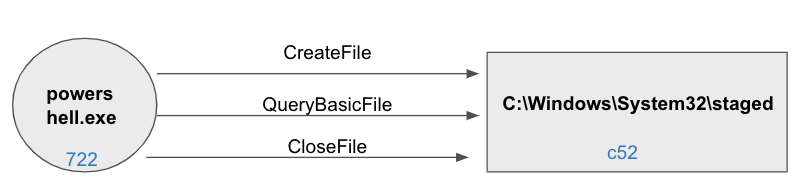
T1036.004\_1f0 Creating W32Time similar named service using sc。該劇本目標是偽照 Windows 的正規服務 W32Time，因此它利用 Service Control Manager(sc) 創建了相似命名的服務 – win32times，該劇本可以達成 T1036.004 Masquerading: Masquerade Task or Service 這個 Technique。



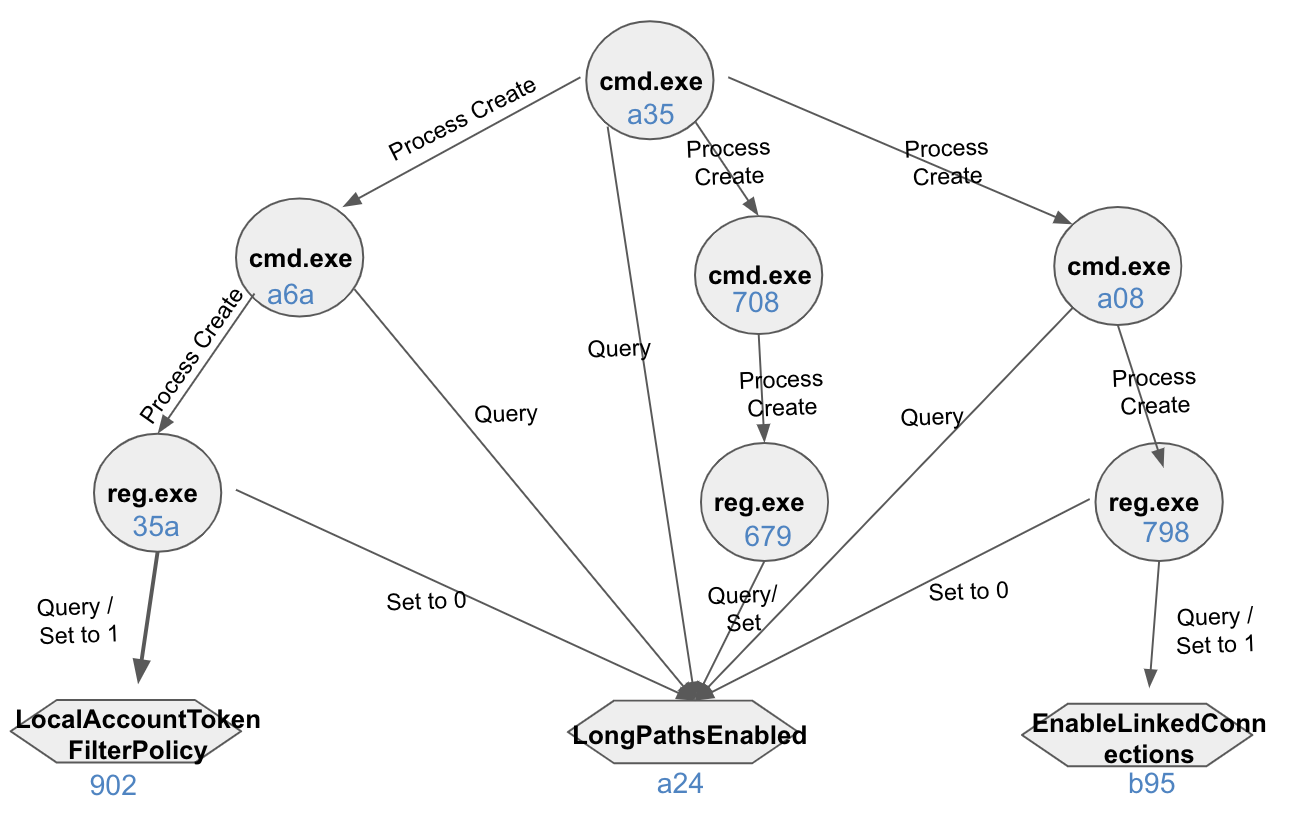
圖D.「兩個」三元組的劇本案例 — T1564.003\_9a2 Hide Artifacts: Hidden Window. 該劇本目標是利用 "-WindowStyle Hidden" 這個參數啟動 PowerShell，通過將 WindowStyle 參數設置為 “hidden” 來隱藏 PowerShell 視窗。執行時，隱藏的 PowerShell 視窗將啟動 calc.exe。該劇本可以達成 T1564.003 Hide Artifacts: Hidden Window 這個Technique。



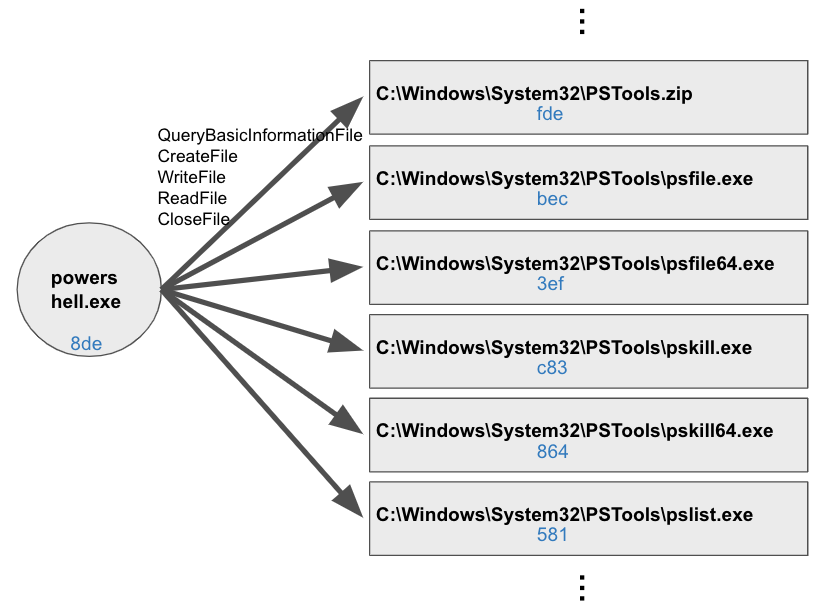
圖E.「三個」三元組的劇本案例 — T1007\_d6b Identify system services. 該劇本利用 tasklist 列出正在執行的程序清單，再利用 Service Control Manager(sc) 將其轉換為預期結果並標準輸出。該劇本可以達成 T1007 System Service Discovery 這個 Technique。



圖F.「三個」三元組的劇本案例 — T1074.001\_646 Create staging directory. 該劇本目標是建立一個可以暫存外洩數據的資料夾。該劇本可以達成 T1074.001 Data Staged: Local Data Staging [ref]這個 Technique。



圖G 是對應到「18個」三元組的劇本案例，該劇本名為 BlackByte Ransomware Registry Changes - CMD，旨在模擬 BlackByte 勒索病毒執行三個步驟：關閉使用者帳戶控制 （UAC）遠端限制、開啟不同權限間網路連接功能、允許長檔名的存在讓檔案可以加密。該劇本可以達成 T1112 Modify Registry 這個 Technique。

圖H 是對應到「150個」三元組的劇本案例，該劇本名為 Download and install PSTools by unzipping the file，該劇本會先下載 PSTools.zip 然後解壓縮，因此系統便會對 PSTools 檔案夾底下的所有檔案都做 CreateFile / WriteFile / ReadFile 等動作，假使 PSTools 檔案夾底下的檔案數量很多，那變會造成大量系統資源的互動，也就產生大量的三元組。

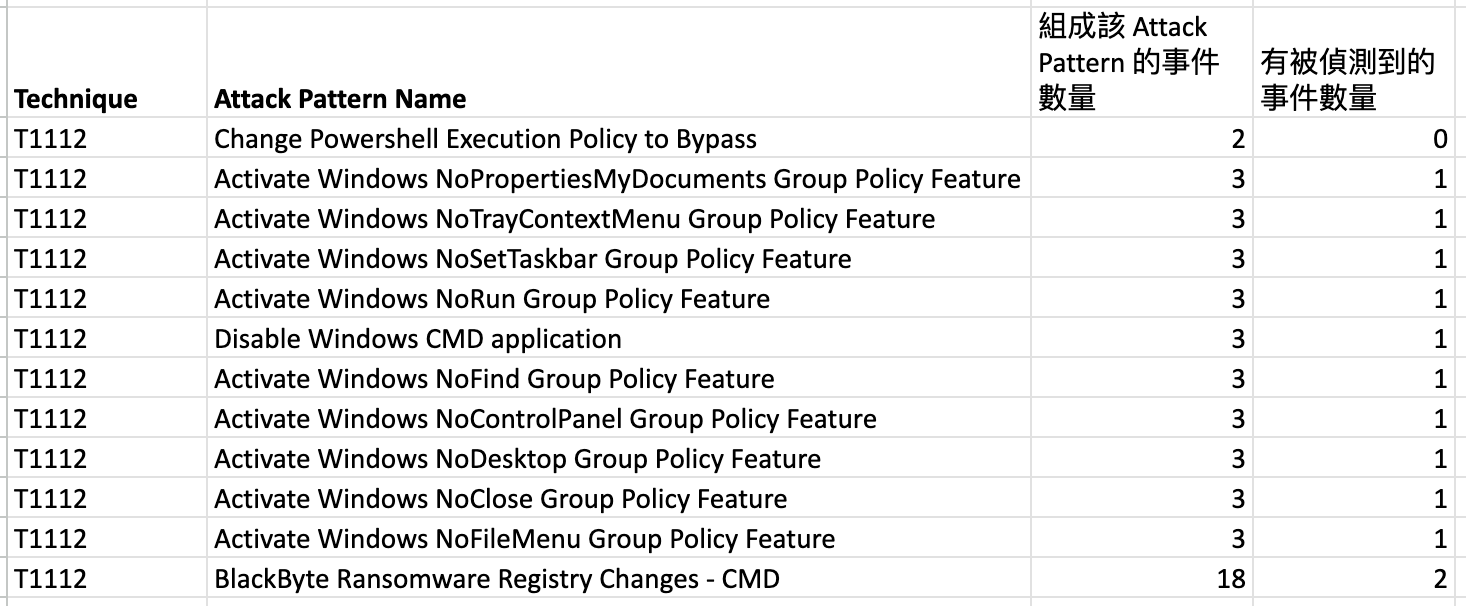
## 【Appendix 2 — TransX Embedding 訓練過程】



圖I. TransE、TransH、TransR 模型訓練過程，各模型的損失函數請見原始論文。

## 

## 【Appendix for Sigma】



（以下我先不看，因為光事前面的部份，你必須先將其內容補齊）

### 【適應性(Lack of Adaptability)】

由這個例子可以觀察到，Sigma Rule 雖然可以偵測到 T1112 其中的 11 個 Attack Patterns，但第一個 Attack Pattern - Change Powershell Execution Policy to Bypass 它沒偵測不到。這有可能是因為 Change Powershell Execution Policy to Bypass 是一個新型的攻擊手法，只要定義 rule 的專家未看過這個攻擊方式，就沒辦法定義這個規則。

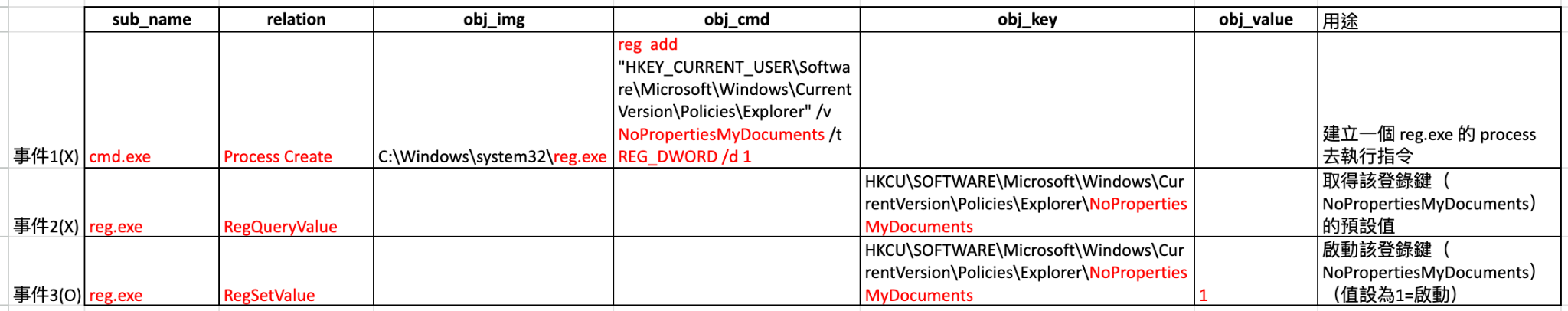
### 【侷限性(Incompleteness)】

第二個觀察是，rule-based 的偵測方式大多只能偵測到該攻擊部分的事件，舉第二個 Attack Pattern: Activate Windows NoPropertiesMyDocuments Group Policy Feature 來說，要完成該手法應該會需要 3 個事件，但 Sigma Rule 卻只偵測出 1 個事件。

去追朔原因的話會發現，

規則是寫說如果登陸鍵是以 *SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoPropertiesMyDocuments* 結尾且執行動作(operation)是 RegSetValue 的話就是 alert。

實際上，如下表所示，事件3有符合這個規則，但事件1和事件2，也就是創建程序以及取得登錄檔的這兩個登錄檔就沒有被偵測到。



## 【Reference】

X. Han, T. Pasquier, and M. Seltzer, “Provenance-based intrusion detection: opportunities and challenges,” in 10th USENIX Work- shop on the Theory and Practice of Provenance (TaPP 2018), 2018.[1]

G. Jenkinson, L. Carata, T. Bytheway, R. Sohan, R. N. Watson, J. Anderson, B. Kidney, A. Strnad, A. Thomas, and G. Neville-Neil, “Applying provenance in {APT} monitoring and analysis: Prac- tical challenges for scalable, efficient and trustworthy distributed provenance,” in 9th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2017), 2017.[2]

X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, “Uni- corn: Runtime provenance-based detector for advanced persistent threats,” arXiv preprint arXiv:2001.01525, 2020.[3]

M. N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. Stoller, and V. Venkatakrishnan, “{SLEUTH}: Real- time attack scenario reconstruction from {COTS} audit data,” in 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 487–504.[4]

S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakr- ishnan, “Holmes: real-time apt detection through correlation of suspicious information flows,” in 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019, pp. 1137–1152.[5]

W. U. Hassan, A. Bates and D. Marino, "Tactical Provenance Analysis for Endpoint Detection and Response Systems," 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2020, pp. 1172-1189, doi: 10.1109/SP40000.2020.00096.[6]

Bibek Bhattarai, H. Howie Huang: “SteinerLog: Prize Collecting the Audit Logs for Threat Hunting on Enterprise Network“. AsiaCCS 2022: 97-108[7]

Zhu, Tiantian et al. “APTSHIELD: A Stable, Efficient and Real-time APT Detection System for Linux Hosts.” ArXiv abs/2112.09008 (2021): n. pag.[8]

T. Yadav and A. Rao, “Technical aspects of cyber kill chain in: International symposium on security in computing and commu- nication, 438–452,” 2015.[9]

M. ATT&CK, “Mitre att&ck,” URL: https://attack. mitre. org, 2021.[10]

X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, “Uni- corn: Runtime provenance-based detector for advanced persistent threats,” arXiv preprint arXiv:2001.01525, 2020.[11]

Abdulellah Alsaheel, et al. "ATLAS: A Sequence-based Learning Approach for Attack Investigation." 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, 2021.[12]

J. Zengy et al., "SHADEWATCHER: Recommendation-guided Cyber Threat Analysis using System Audit Records," 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2022, pp. 489-506, doi: 10.1109/SP46214.2022.9833669.[13]

Fan Yang, Jiacen Xu, Chunlin Xiong, Zhou Li, Kehuan Zhang. “PROGRAPHER: An Anomaly Detection System Based on Provenance Graph Embedding.” Usenix, 2023.[14]

F., Roth, and Patzke T. “Sigma: generic signature format for SIEM systems.” GitHub, 2019, https://github.com/SigmaHQ/sigma/tree/master. Accessed 27 July 2023. [15]

MITRE Corporation. (2022). CALDERA: A Scalable, Automated Adversary Emulation Platform (Version 4.1.0) [Computer software]. <https://github.com/mitre/caldera> [16]

MITRE Corporation. (2022). CALDERA: Basic Usage — Caldera Documentation. caldera.readthedocs.io/en/stable/Basic-Usage.html. Accessed 1 Aug. 2023.[17]

Markruss. “Process Monitor - Sysinternals.” Microsoft Learn, 9 Mar. 2023, learn.microsoft.com/en-us/sysinternals/downloads/procmon. [18]

Caldera 攻擊報告（還在找官方檔案，之後確定沒有會補上截圖）[19]

Wikipedia contributors. “Google Knowledge Graph.” Wikipedia, July 2023, en.wikipedia.org/wiki/Google\_Knowledge\_Graph. [20]

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: a nucleus for a web of open data. In Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference (ISWC'07/ASWC'07). Springer-Verlag, Berlin, Heidelberg, 722–735. [21]

Cuckoo Sandbox - Automated Malware Analysis. 19 June 2019, cuckoosandbox.org. [22]

Cuckoo Sandbox 惡意程式分析報告（還在找官方檔案，之後確定沒有會補上截圖） [23]

Welcome to Faker’s Documentation! — Faker 18.13.0 Documentation. faker.readthedocs.io/en/master. [24]

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko,

“Translating embeddings for modeling multi-relational data,” in NIPS, 2013, pp. 2787–2795. [25]

Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in AAAI, 2014, pp. 1112–1119.[26]

Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, “Learning entity and relation embeddings for knowledge graph completion,” in AAAI, 2015, pp. 2181–2187.[27]

Ehsan Aghaei, Xi Niu, Waseem Shadid, & Ehab Al-Shaer. (2022). SecureBERT: A Domain-Specific Language Model for Cybersecurity. [28]

Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805

(2018) [29]

1. Only downloaded rules that is under these 5 folders:

   * rules/windows/file
   * rules/windows/registry
   * rules/windows/network\_connection
   * rules/windows/process\_access
   * rules/windows/process\_creation

   [↑](#footnote-ref-0)
2. 已知三元組是指在知識圖譜中已經存在的事實，也就是可以確定是正確的三元組。而負例三元組（Corrupted Triplets）是在已知的三元組之外隨機生成的，這些負例三元組是不符合知識圖譜的事實的，也就是錯誤的三元組。 [↑](#footnote-ref-1)
3. 多義性 (Ambiguity)：指一個實體或關係擁有多個不同的含義或解釋。細節請見原始論文。 [↑](#footnote-ref-2)
4. 多重關係 (Multiple Relations)：指一個實體和另一個實體之間存在多個不同的關係。細節請見原始論文。 [↑](#footnote-ref-3)