以資訊安全維運中心為基礎之雲端平台事件處理系統

黃瓊瑩[1]、孫明功[1]、陳嘉玫[2]、龍志雄[2]、賴谷鑫[3]
宏碁電子化資訊管理中心[1]、國立中山大學資管系[2]、臺灣警察專科學校科技偵查科[3]
C.Y.Huang@acer.com、Morgan.Sun@acer.com、cchen@mail.nsysu.edu.tw、
jeric701028@hotmail.com、ghlai@cc.tpa.edu.tw

摘要

近年來攻擊型態開始逐漸轉變，不同以往傳統隨機的攻擊手法，現在多數攻擊型態是以針對特定攻擊目標的攻擊模式，常稱為目標式攻擊(Targeted Attack)或進階持續性滲透攻擊(Advanced Persistent Threat, APT)。此類攻擊者大多屬於具有高度知識與豐富資源的駭客組織，使用如社交工程技術、客製化惡意程式、零時差攻擊等進階手法來當成入侵的手段，通常具有針對性，會選擇政府或企業中重要的單位作為其攻擊目標。

為了防禦日新月異的 APT 攻擊手法，以安全性資訊與事件管理(Security Information and Event Management, SIEM）為基礎並建置資訊安全維運中心（Security Operation Center, SOC）之解決方案應運而生。透過前端日誌收集器（Agents）將各種不同類型之資安設備如防火牆、入侵偵測系統、誘捕系統、防毒軟體與資料外洩防禦系統等所產出日誌正規化，並導入分析平台以進行關聯分析監控與產生即時告警。另這些資安設備拋送出的日誌，亦可透過大數據平台挖掘、深度分析與產出數據統計報表，進一步找出潛在的資訊安全風險危機。

SIEM 藉由整合以及分析組織內之資訊安全設備日誌以偵測 APT 攻擊，然而 SOC 在處理資訊安全的紀錄檔時面臨第一個挑戰即為如何將異質資料格式進行正規化，進而使後端的分析平台可以根據這些資訊分析、告警以及產出報表。有鑑於此，本研究開發一套智慧型記錄檔剖析系統（Intelligent Parsing System）透過該系統可以針對不同日誌快速產生高品質之正規表示式。透過本研究所開發的系統可以快速地將異質日誌檔整合至單一的 SOC 系統中。

關鍵詞：進階持續性滲透攻擊、安全性資訊與事件管理、資訊安全維運中心

A Cloud Based Security Event Management System in SOC Environment

Chiung-Ying Huang[1], Ming-Kung Sun[1], Chia-Mei Chen[2], Jhih-Syong Long [2], Gu-Hsin Lai[3]
Acer CyberCenter Services, Inc[1], Department of Information Management, National Sun Yat Sen University[2], Department of Technology Crime Investigation, Taiwan Police College[3]
C.Y.Huang@acer.com, Morgan.Sun@acer.com, cchen@mail.nsysu.edu.tw,
jeric701028@hotmail.com, ghlai@cc.tpa.edu.tw

Abstract

Conventionally, there is not specific targets when cyber attacks happen. However, increasing portion of attacks aim at specific targets in recent years. Emerging targeted attacks are referred to as Targeted Attack or Advanced Persistent Threat (APT). Such attacks typically aim at government agencies or critical departments in enterprises, and are usually implemented by knowledgeable and resourceful attackers by utilizing advanced invasive methods such as social engineering techniques, customized attacking programs, and zero-day attacks.

To defend against rapid changing APT attacks, Security Operation Center (SOC) that is based on Security Information and Event Management (SIEM) is thus developed. Several log collection agents collect and normalize logs from different security devices like firewall, IDS, honeypot, anti-virus software, NDLP and so on. SIEM could use these log to perform correlation analysis to monitor and generate real-time warnings. Besides, log data collected from these security equipment is suitable for big data excavation, deep analyses for reports generation, thus unveil potential risk.

SIEM can detect APT attacks by integrating and analyzing log in the organizations. However, one of the promptest challenge SOC faces is how to normalize data of heterogeneous formats so that back-end analytic platforms can analyze information, alarm and generate reports accordingly. Therefore, this research develops an intelligent parsing system to generate high quality normalization equations for different types of log. The proposed system could integrate heterogeneous log files into a central SOC system

Keywords: APT, SIEM, SOC

壹、 前言

隨著網際網路的蓬勃發展，相關資訊科技的日益進步，促使各產業的蓬勃發展，而網路設備的普及與網路的便利使用，更觸及人們生活中的每一個部分，但也直接提高網路犯罪對一般使用者與企業組織所造成的危害[8]。此外，病毒、蠕蟲、特洛伊木馬、間諜程式或是一些其他社交工程攻擊，近年來成為網際網路日趨嚴重之威脅，上述這些程式可統稱為惡意軟體[12][3]。在網際網路的環境中，惡意軟體帶來的危害可由以下數據得知：知名防毒軟體廠商賽門鐵克（Symantec Inc.）之調查報告指出，惡意軟體的數量成長幅度驚人，一年已偵測到超過 4 億次之惡意軟體攻擊[2]。而面對這些數量龐大的入侵攻擊，網路管理者會建置不同的防衛與偵測系統，如防火牆、入侵偵測系統、垃圾郵件過濾器、代理伺服器或誘捕系統（Honeypot）等，以阻擋各式各樣的攻擊行為。根據

研究機構 Ponemon 之分析報告指出，美國企業在 2013 年遭受網路攻擊的損失將近 1160 萬美元，相較於 2012 年成長 26%，平均每週遭受到網路攻擊成功次數高達 122 次，而平均每次解決一件網路攻擊必須花費 32 天，甚至最長高可達到 65 天，在這 32 天期間，企業所涉及的平均損失為 1,035,769 美金，較去年平均 24 天的解決時間所估計的 591,780 美金平均損失增加 55%[7] 。除此之外，從研究機構 Wildlist 於 2008 年到 2013 年期間研究報告得知，每月平均仍然有高達 604 筆惡意軟體威脅網際網路[11]。由此可見，網路的攻擊已造成極大的損失且有日趨嚴重之傾向。

近年來，攻擊型態開始逐漸轉變，為了特定攻擊目的與產生更大的效果，針對特定攻擊目標的攻擊模式，逐漸形成一股不可忽略之趨勢，並出現一種名為進階持續性滲透攻擊（Advanced Persistent Threat, APT）之手法，攻擊者本身具備高度電腦網路背景知識，常以組織的方式運作，因此擁有充分的資源可進行系統化的部署，目標通常具有針對性，大多是政府和企業中重要的設備或敏感資訊系統。APT 攻擊的手法與媒介呈現多元化且經常根據目標以進而高度客製化，與利用零時差漏洞資訊，整個部署的時間長達幾個月甚至數年。由於此類攻擊手段主要目的在於竊取所需要的特定資訊，如國家安全或是商業上的機密等，耗費時間較長且高度隱密，因此巨幅增加防禦困難度。由於傳統的入侵偵測系統採用信譽評等或特徵比對演算法邏輯，雖然能阻擋過去常見的攻擊，但進階持續性滲透攻擊，其所採用的手段偏向未知的特徵，過去的入侵防禦系統變得難以防護，等到成功滲透之後，透過僵屍網路的遠端控制，讓攻擊者隱匿於網路中，更讓 APT 攻擊的偵測難上加難。根據資訊安全公司 McAfee 2011 年所公布的一份資訊安全調查報告顯示，美國是 APT 最大的受害國家，其次為加拿大，南韓和臺灣並列第三名，而其中又以政府和高科技產業為主[1]，到了 2014 年統計，非政府單位受害比例提高至將近 1/4，遭 APT 攻擊鎖定的產業包括電子業產品製造鏈、金融與傳統醫療等產業，並有向中小企業擴大的趨勢，因此如何偵測 APT 攻擊已是刻不容緩的議題[13]。

為了保護組織內部免於駭客攻擊，各式各樣的資訊安全設備被佈署於組織內部，常見如防火牆、入侵偵測系統、防毒軟體、誘捕系統與資料外洩防禦系統（DLP)等。然而上述設備皆旨在提供特定的功能，如果要偵測如 APT 攻擊或殭屍網路（Botnet），需具備同時關聯數種日誌檔以進行分析之能力，因此以安全性資訊與事件管理（Security Information and Event Management, SIEM）為基礎並建置資訊安全維運中心（Security Operation Center, SOC）之解決方案應運而生。相較於傳統的單形態設備分析，SIEM 可集中管理組織內部資訊安全設備與重要主機的日誌檔案（Log），關聯多種異質設備以進行綜合情報分析，透過資訊安全監控維運中心 24 小時不間斷持續監控，可偵測任何可疑的網路行為與發出即時告警（Alert），以提早發覺潛伏的資安威脅並避免可能造成的損失。

隨著更大範圍的資訊安全設備佈署，與跨設備關聯分析，資訊安全監控維運中心面臨以下兩個問題：異質性資料處理、以及如何分析龐大的資料量。不同類型之資安設備之日誌資料欄位通常不盡相同，輸出的資料格式亦五花八門，更有甚者，同一種資安設

備但不同韌體版本可能產生出不同格式紀錄檔。SIEM 面臨第一個挑戰即為如何正規化各種異質資料格式，以下說明日誌正規化之重要性：

1. 截取日誌中關鍵的欄位，進而使 SOC 分析平台可以根據這些資訊分析、告警以及產出分析報告。
2. 同類但不同廠商之設備日誌欄位常存在不同表達方式，如 Deny、Block 或 Prohibit 其實代表相同連線意義，經過統一正規化為 Failure 可使日誌格式具備一致性，並方便集中控管。
3. 相較於其它的分析工具平台，SIEM 在整個日誌的生命週期中從一開始即考慮到其特性，強制要求將日誌進行正規化，去除不必要的資訊以節省儲存空間，賦予其進一步深度分析之價值，因此正規化是 SIEM 平台中十分重要的工作。

由於科技日新月異，即使是專業的 SIEM 系統也無法一舉處理市場上所有的資訊安全設備日誌，因而提升了收集日誌的困難度，此時客製化解析（Customized Parsing）成為無可避免的程序，此工作一般仰賴資安工程師負責，以下將說明過程中經常遭遇的困難：

1. 使用客製化解析器可以解決資料格式與結構整合問題，然而藉由專業人力進行大量客製化工作對於 SOC 將造成額外的負擔，另無可避免將衍生出人工作業緩慢與除錯不易等問題。
2. 在資安問題層出不窮的今日，各式各樣的設備具備不同的專業功能，其產出的日誌格式自然五花八門，即使是受過專業訓練之資安工程師也很難同時熟悉各種不同日誌解析方法。
3. 手動撰寫的解析器（Parser）其執行效率將隨著撰寫者專業素養不同而產生效率高低之分。舉例下圖一為防火牆系統的紀錄檔，可分別經由圖二、圖三以及圖四的正規表示法來正確解析，但彼此間卻有著不小的效能上差異，圖四的撰寫方式明顯勝於其餘兩者，可大幅提升日誌收集與正規化的效能。

```
(2016-07-01T07:20:10) firewall: Deny 6-Aruba-WiFi Firebox 44 udp 20 37 xxx.xxx.xxx.xxx xxx.xxx.xxx.xxx 16387 16403  (Unhandled External Packet-00)
```

圖一、防火牆紀錄檔範例

```
Regex=\((\d+\-\d+\-\d+T\d\d:\d\d:\d\d)\) firewall: (\S+) (\d+\-Aruba\-WiFi Firebox) \d+ udp \d+ \d+ (\d+\.\d+\.\d+\.\d+) (\d+\.\d+\.\d+\.\d+) (\d+) (\d+)  \((.*)\)
```

圖二、正規表示式 1

```
Regex=\((\d{4}-\d{2}-\d{2}\S\d{2}:\d{2}:\d{2})\)\sfirewall:\s[Allow|Deny]+\s(\S+)\s\S+\s\d+\s(\S+)\s\d+\s\d+\s([\d{1,3}.]+)\s([\d{1,3}.]+)\s(\d+)\s(\d+)\s+\((.*)\)
```

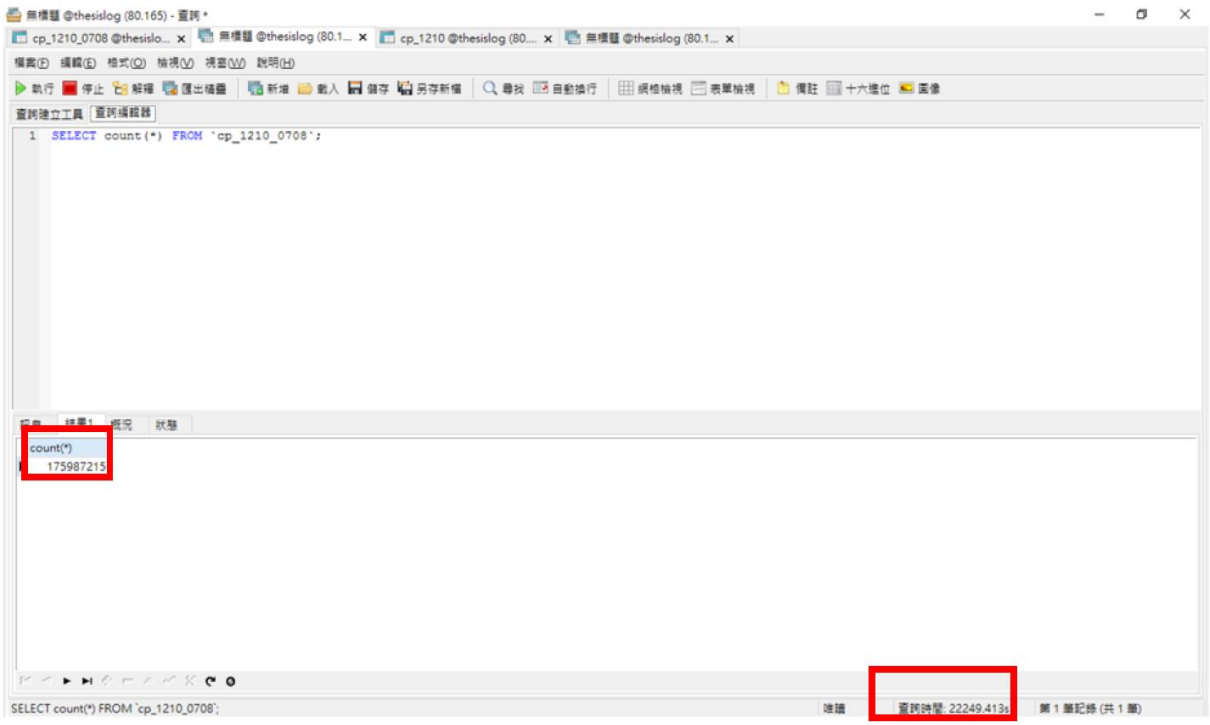
圖三、正規表示式 2

```
Regex=\(([^\s]+)\)\s([^\s]+):\s([^\s]+)\s([^\s]+)\s[^\s]+\s[^\s]+\s([^\s]+)\s([^\s]+)\s([^\s]+)\s([^\s]+)\s([^\s]+)\s([^\s]+)\s([^\s]+)[^\(]+\(([^\)]+)\)
```

圖四、正規表示式 3

縱觀上述問題，資訊安全維運中心需要一個有系統、自動化的方式針對異質設備之日誌檔案產生出一致且高品質的解析器，以提供 SOC 作為事件分析的基礎，進而提高 APT 或是目標式攻擊的偵測率。

另一方面，當收集了大量高品質的資安設備日誌後，資料的儲存成為另一個挑戰。過去傳統的關聯式資料庫（Relational Database Management System, RDBMS）逐漸無力處理如此大量的資料，為了探討關聯式資料庫是否適用於資訊安全監控維運中心，本研究將真實收集的一天上班時間的資料（175,987,215 筆，約 133G）匯入關聯式資料庫並測試其效能，此實驗於在 6 顆 CPU 與記憶體 20G 的環境下進行簡單的總筆數查詢運算，結果如下圖五所示，共耗費 22, 249 秒以獲得查詢結果。



圖五、傳統 RDBMS 效率模擬實驗

由以上實驗顯示，傳統關聯式資料庫光是要計算 8 小時的事件數量就需要超過 6 小時的時間，其揭露出 RDBMS 無法應付像 APT 此類時程長達半年甚至更久的攻擊手法。本論文將針對以上兩個問題開發出一套以資訊安全維運中心為基礎之雲端平台事件處理系統，SOC 之維運人員將可以快速的整合日誌檔，並在平台上將大量紀錄檔關聯並整

合於雲端架構中以利未來大數據以及長時間軸之紀錄檔分析。

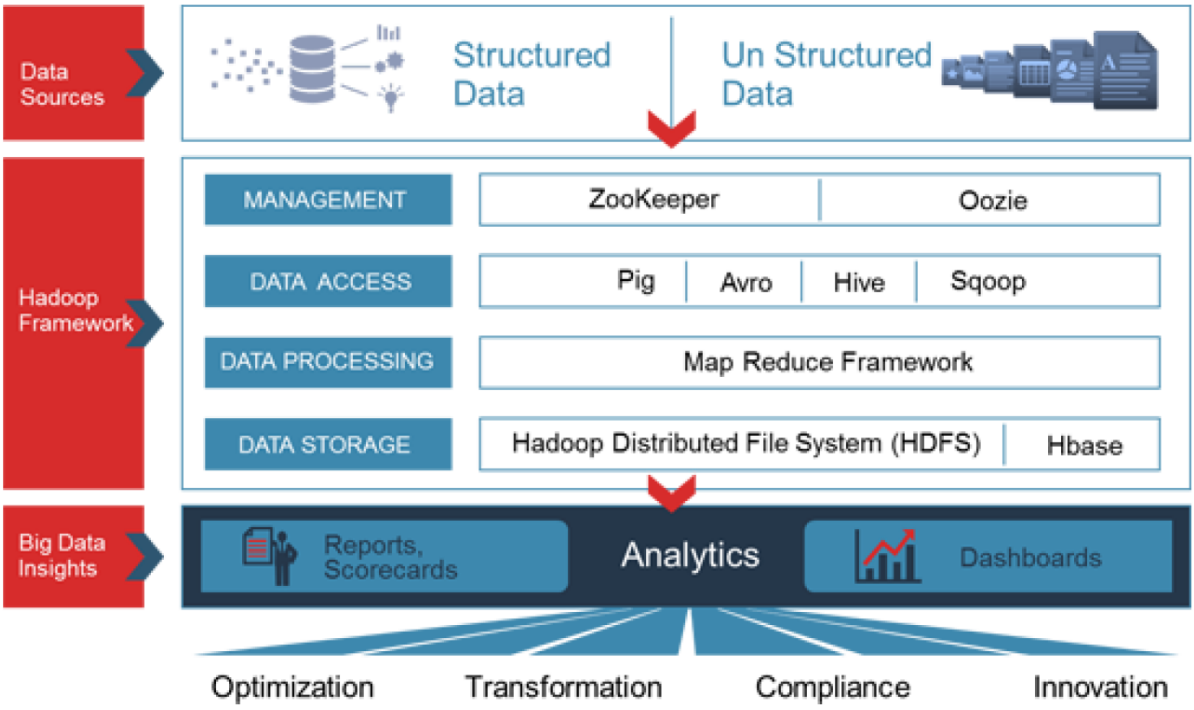
貳、 文獻探討

隨著各種資安設備的佈署，各種資安設備產生出大量的異質的紀錄檔，而許多專家學者以人工智慧或是機器學習的方式偵測網路上的惡意行為。當原始日誌記錄檔逐漸增加時，各種資安設備所儲存的資料欄位，以及輸出資料的格式不盡相同，卻也間接導致分析的問題與資料處理的難度。解析器（Parser）的使用與應用可以解決資料欄位的差異，以及作為後續關聯分析的基礎，Pinjia He 等學者指出[4]在日誌解析的過程中，原始資料（Raw Log）可以被解析為結構化的資料（Structured Logs）與該筆資料所包含的事件內容（Log Events），且對於後續需要進行日誌探勘的處理來說，解析的過程是極為重要的，因為準確度夠高時且經過分析處理過的資料才會有效，數據及圖表所呈現出來意義也才不會有所偏差。

然而不論用何種分析模式，都必須面對到大量資料處理的問題，處理以及分析巨量資料已然成為多數研究的關鍵，而如何從巨量資料中找到需要且重要的資料，更是迫於眉睫的問題。若利用傳統關聯式資料庫（RDBMS）的架構，已經無法迅速對巨量資料進行分析，傳統式關聯式資料庫處理的資料大小約為 Gigabytes，而透過 Hadoop 的 MapReduce 技術可以處理約 Petabytes 等級的運算[5]。由此可見，當企業在面臨巨量紀錄檔以及目標式攻擊分析挑戰的同時，勢必得使用 Hadoop 等大數據分析技術來著手處理。Hadoop 是以分散式檔案系統（Hadoop Distributed File System, HDFS）和由 Google 所開放原始碼的 MapReduce 為核心所組成，讓使用者能不須了解分散式檔案系統底層細部運作原理，即可開發平行應用程式，來處理大量資料的軟體平台。其基本架構如圖六所示[6]，使用分散式檔案系統的好處是具有高容錯、高伸縮性，可以快速部屬在較低價的硬體設備上，而 Hadoop 的基本運作方式即是以 MapReduce 框架將檔案拆成數個區塊，再分給叢集上的節點執行。

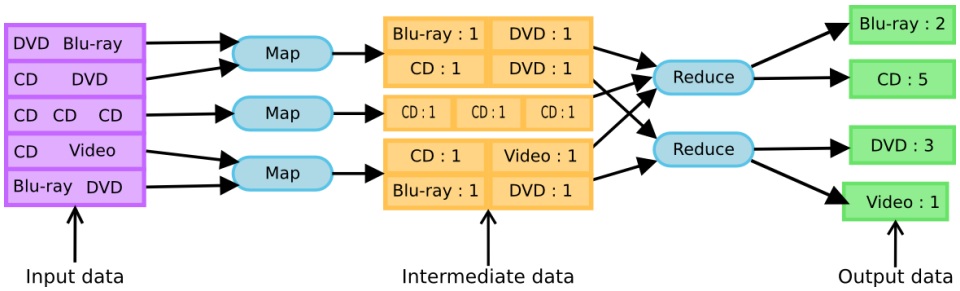
Hadoop 是以 Java 所開發，可不受限於作業系統，目前在科學研究與產業環境中已被廣泛使用，其特色如下：

(1) 巨量：擁有儲存與處理大量資料的能力。
(2) 經濟：能用於一般個人電腦所架設的叢集環境。
(3) 效率：透過分散式檔案系統的幫助下，得以得到快速的回應。
(4) 可靠：當節點發生錯誤時，可以即時取的備份資料以及重新佈署運算資源。



圖六、Big Data 環境中 Hadoop 架構圖

當資料量小的時候，可以將檔案儲存在一顆或是多顆硬碟上面，來對其進行相關的運算或是操作；而一旦資料量大的時候，傳統式的架構便無法滿足如此需求，同時在多顆硬碟的情境中，也難以對資料進行管理[10]。在 Hadoop 的整個生態中，不僅提供名為 Hadoop Distributed File System（HDFS）的分散式檔案系統架構，也提供了 MapReduce 分散式運算的框架，讓使用者可以輕易地透過使用 map 和 reduce 函式，來對 HDFS 內的檔案進行分散式運算，以達到降低計算時間的功效。下圖七 為 MapReduce 的運作範例說明[10]。

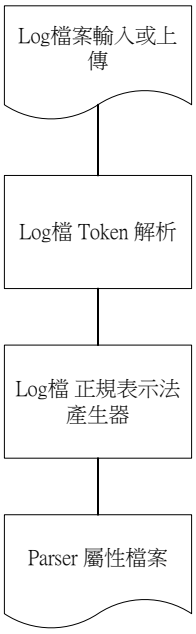


圖七、MapReduce 運算範例

Google 所提出的 Map 和 Reduce 兩個函數編程的分散式運算，讓使用者易於開發使用，不必擔心 Socket 的處理問題，並很容易即可在大量的 CPU 上交互運算。Map 函數中資料輸入是一組 Key-Value 序對，輸出則為另一組 Intermediate Key-Value 序對。Reduce 函數則負責針對所有關聯的 Intermediate key 做合併，並產出一組 Key-Value 作為輸出結果。值得注意的是，雖然 MapReduce 提供了一個優異的分散式系統運算架構，但其實際上運作時，會將暫存值存放到硬碟中，以檔案方式進行儲存，因此硬碟的 I/O 存取速度將成為整個運算過程的效能瓶頸。為了改善此狀況，本研究將使用 Spark 模組作為資料處理的技術。Spark 主要是透過一個以分散式記憶體為主的計算模型，來進行巨量資料的運算，相對於 Hadoop 的 MapReduce 在工作執行後，將暫存值存放到硬碟中，以檔案方式進行儲存，Spark 使用了記憶體內運算技術，能在資料尚未寫入硬碟時即在記憶體內分析運算，以達到大幅提高計算效率的目的。Spark 不僅改進了 Hadoop 用的傳統 MapReduce，也提供了豐富而且易用的 API，開發者可以使用自己熟悉的程式語言來進行開發，其中包括 Java、 Scala 與 Python 等多樣化之程式語法，透過此架構之雲端平台將可以處理與分析數量龐大的資訊安全紀錄檔。

參、 系統架構

本研究的主要目標為設計一套智慧型 Parser 產生器，用本系統所產生的 Parser 可以用來解決以及整合異質 log 檔案。智慧型 Parser 產生器可以將異質 log 檔完美的整合於 SOC 系統內用以偵測更多，更複雜的網路攻擊。由於資訊安全監控維運中心所收集日誌檔來自各種不同的設備，因此有必要將其轉成固定通用的格式，另在轉換過程中將出現另一個問題，在企業中隨時因為添購產品或是產品的更新，版本的不同導致紀錄檔格式的變化（舉例而言，Windows Active Directory 紀錄檔其 Windows 2000 以及 Windows Server 2008 格式是不一樣的）。因此該如何因應因為添購資訊安全設備或是產品升級所產生新格式紀錄檔成為資訊安全維運中心一個重要議題。如何讓管理者快速的針對新格式的紀錄檔產生一致且高品質解析器 將是資訊安全監控維運中心重要的工作之一。因此在本研究將建立一個根據個別紀錄檔格式自動化產生 Parser 設定檔的系統。下圖八為本研究系統架構圖

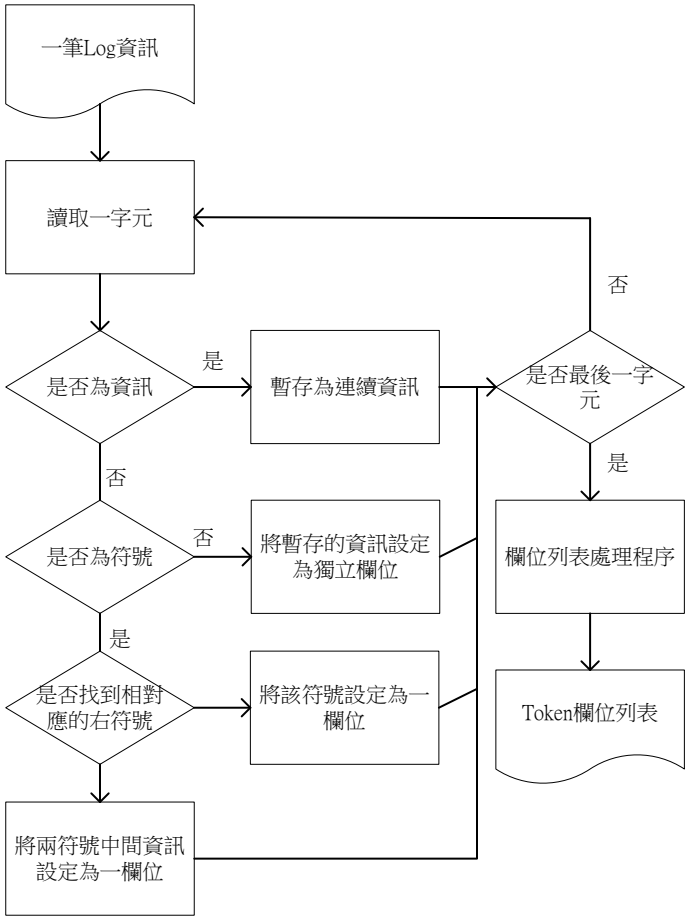


圖八、系統架構圖

本研究所設計的系統主要分為兩個模組，第一個模組為 Log 檔 Token 解析模組。該模組主要功能是將輸入的 Log 檔案拆解為 Token(欄位)。本系統提供欄位選擇、變數設定以及 sub-message 功能。透過這些功能可以將紀錄檔更完整的整合至 SOC 系統中，其操作細節將於下一章節系統展示說明。而本系統第二的模組則是正規表示法產生器，透過第一個模組所切割的 Token 產生出對應的正規表示法。而本系統最後所產生的為 Parser 屬性檔案。SOC 系統可以讀取該屬性檔案直接對 Log 檔案做剖析。

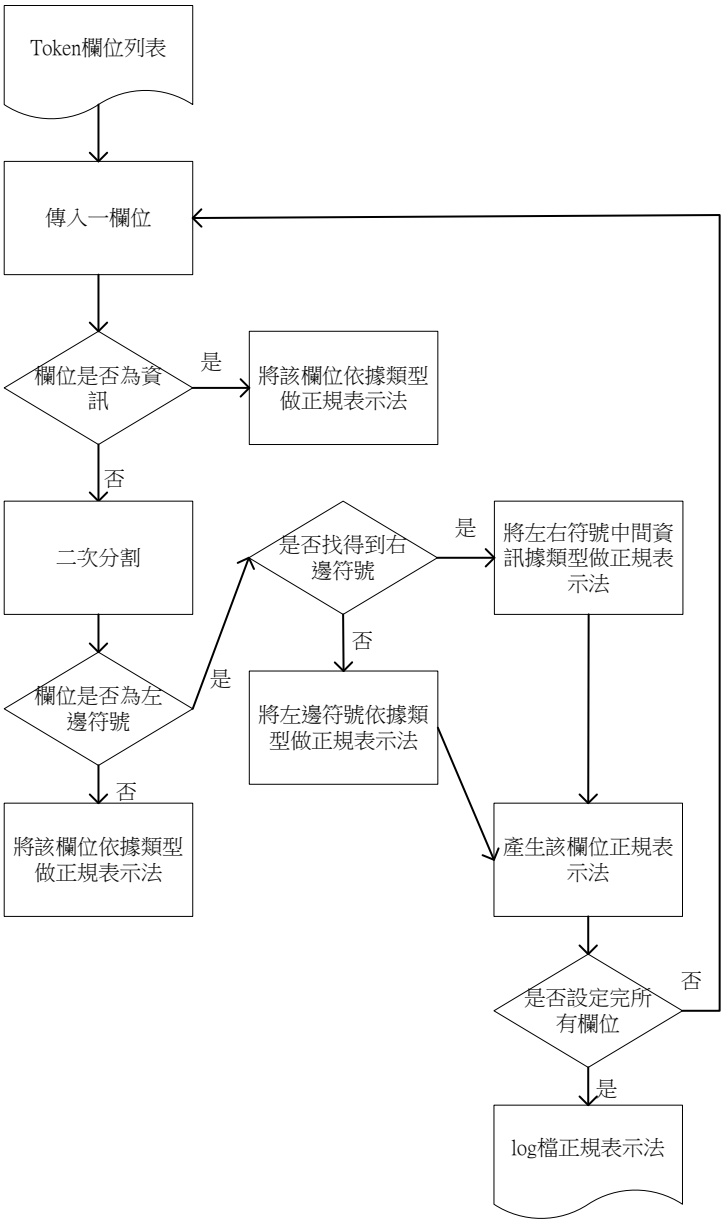
本研究的 Parser 產生器採用 LL Parser 的概念做為演算法的基礎。LL Parser 是一種處理與上下文無關、的自頂向下的 Parser 演算法。LL Parser 的特性為左（Left）到右處理輸入，再對句型執行最左推導出語法樹（Left derivation）。本研究的 Parser 產生器將常見的符號做出區隔，來進行第一階段的自動解讀並切割出有意義的資料，其中包含一些人類語言的使用習慣來模擬資訊人員看到 Log 檔案時的解讀方式。這階段的結果會將自動解讀完成的資料設定成各個不同的欄位並以表格的方式顯示在畫面上。而接下來，資訊人員即可以自主進行更詳細的修訂，例如：設定欄位名稱及型態、設定每個欄位所需要的屬性等等。若自動解讀的欄位不合要求或是需要更加精細，針對每個欄位都能夠自動進行二次解讀，甚至是將欄位顯示或是刪除。除此之外，本研究的正規表示法產生模組，能夠自動產生對應到每筆 Log 資料的正規表示法。依據資訊人員設定完成的型態，正規表示法模組能夠更加精確的產生相對應的結果，來提供 SOC 使用。當然也提供資訊人員手動修改的功能，並在修改完成後檢查正規表示法是否與 Log 資料吻合，以防一些人為疏失。

下圖九為本系統的 Token 解析模組運作流程



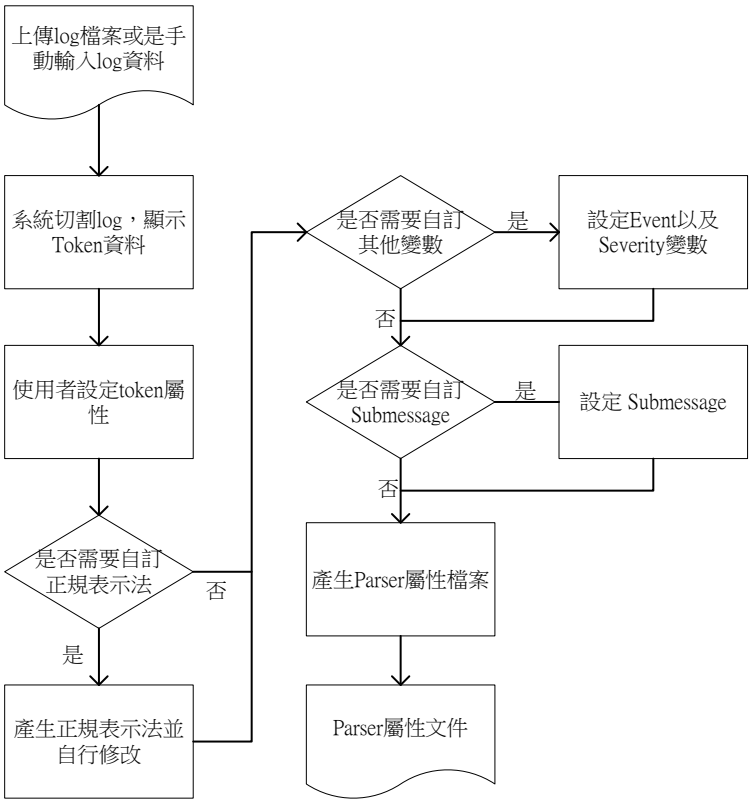
圖九、Token 解析模組運作流程圖

本系統將輸入的字元分為資訊以及符號。資訊類型分為資訊與符號，資訊為中英數以及小數點、反斜線、底線等等而符號則為剩餘的其他。本系統以類似 LL 剖析的方式區分出 Token 以及包覆在 Token 外側的符號。本系統將處理特殊規則的 Token。特殊規則的部分為遇到冒號以及逗號時這兩種情況。當遇到冒號時表示後面的資訊為前方資訊的內容，故將空格暫時加入為資訊。當遇到逗號時並且存在冒號的情況下判斷後方字母是否為大寫開頭，是的話表示為新的資訊。本模組著重於 Log 檔中 Token 的剖析。而產生出 Log 檔的 token 列表後，本系統根據每個 token 產生出對應的正規表示式。正規表示式的產生模組流程如下圖十。



圖十、正規表示法產生器運作流程圖

為了產生好的正規表示式，本系統依據資料格式產生正規表示式。根據常見的 log 形態本研究定義出 10 種資料型態，分別為 String、Date、Time、TimeStamp、IPAddress、IPv6Address、Integer、Long、MacAddress、RegexToken。將根據不同型態產出不同的正規表示式。透過 log 檔 token 剖析以及 token 正規表示式，本研究所提出的系統可以有效率且快速的產生優良的正規表示式。而下圖十一則為本系統的操作流程圖



圖十一、系統操作流程圖

而本研究之正規表示法虛擬碼如下圖十二所示。

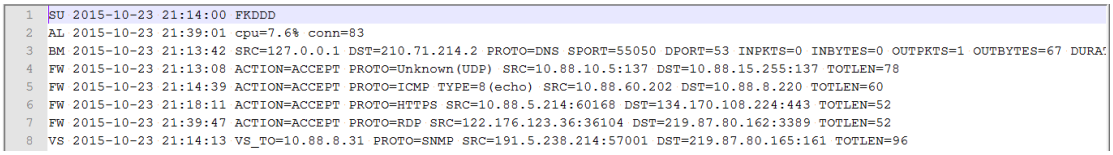
```
DEFINE parenthese as " ' ( ) [ ] { } < >
String FUNCTION RegexGenerate(TokenData[])
    SET Regex TO empty string
    SET Delimiter TO empty string
    FOR i = 0 TO TokenData.count INCREMENT 1
        IF there is a complete parenthese column THEN
            IF TokenData[i + 1] is message THEN
                IF Delimiter is not empty AND doesn't contains TokenData[i] THEN
                    IF TokenData[i].Information is " OR ' THEN
                        ADD [^TokenData[i].Information]+ TO Regex
                    ELSE
                        ADD [^\TokenData`i].Information]+ TO Regex
                    END IF
                ELSE
                    ADD DelimiterRegexGenerate(Delimiter) TO Regex
                END IF
                SET Delimiter TO empty string
                IF TokenData[i + 1].Information contains TokenData[i + 2].Information THEN
                    IF TokenData[i + 1].Information contains whitespace character THEN
                        IF TokenData[i].Information is " OR ' THEN
                            ADD TokenData[i].Information(.*)TokenData[i + 2].Information TO Regex
                        ELSE
                            ADD \TokenData[i].Information(.*)\TokenData[i + 2].Information TO Regex
                        END IF
                    ELSE
                        IF TokenData[i].Information is " OR ' THEN
                            ADD TokenData[i].Information([^\s]+)TokenData[i + 2].Information TO Regex
                        ELSE
                            ADD \TokenData[i].Information([^\s]+)\TokenData[i + 2].Information TO Regex
                        END IF
                    END IF
                ELSE
```

```
                    IF TokenData[i].Information is '' OR ' THEN
                        ADD TokenData[i].Information([^TokenData[i + 2].Information]+)TokenData[i + 2].Information TO Regex
                    ELSE
                        ADD \TokenData[i].Information([^TokenData[i + 2].Information]+)\TokenData[i + 2].Information TO Regex
                    END IF
                END IF
                ADD 2 TO i
                CONTINUE
            END IF
        END IF
        CASE TokenData[i].Type OF
            :
                ADD TokenData[i].Information TO Delimiter
            String:
                ADD DelimiterRegexGenerate(Delimiter) TO Regex
                SET Delimiter TO empty string
                IF TokenData[i].Information contains whitespace character THEN
                    IF TokenData[i].Information matches regular_expression([A-Za-z][^=]+=) THEN
                        ADD ([A-Za-z][^=]+=) TO Regex
                    ELSE IF TokenData[i].Information matches regular_expression([A-Za-z][^:]+:) THEN
                        ADD ([A-Za-z][^:]+:) TO Regex
                    ELSE
                        ADD (.*) TO Regex
                    END IF
                ELSE
                    ADD ([^\s]+) TO Regex
                END IF
            TimeStamp:
                ADD DelimiterRegexGenerate(Delimiter) TO Regex
                SET Delimiter TO empty string
                IF TokenData[i].Information contains whitespace character THEN
                    IF TokenData[i].Information matches regular_expression([^\s]+\s[^\s]+) THEN
                        ADD ([^\s]+\s[^\s]+) TO Regex
                    ELSE IF TokenData[i].Information matches regular_expression([^\s]+\s[^\s]+\s[^\s]+) THEN
                        ADD ([^\s]+\s[^\s]+\s[^\s]+) TO Regex
                    ELSE
                        ADD (.*) TO Regex
                    END IF
                ELSE
                    ADD ([^\s]+) TO Regex
                END IF
            Date, Time, IPAddress, IPv6Address, Integer, Long, MacAddress, RegexToken:
                ADD DelimiterRegexGenerate(Delimiter) TO Regex
                SET Delimiter TO empty string
                IF TokenData[i].Information contains whitespace character THEN
                    ADD (.*) TO Regex
                ELSE
                    ADD ([^\s]+) TO Regex
                END IF
        END CASE
    END FOR
    IF Delimiter is not empty AND the end OF Regex is not .* THEN
        ADD .* TO Regex
    END IF
    REPLACE \ OF Regex TO \\
    REPLACE mutiple \r\n OF Regex TO single [\\s]+
    RETURN Regex
END FUNCTION
```

圖十二、自動化正規表示式虛擬碼

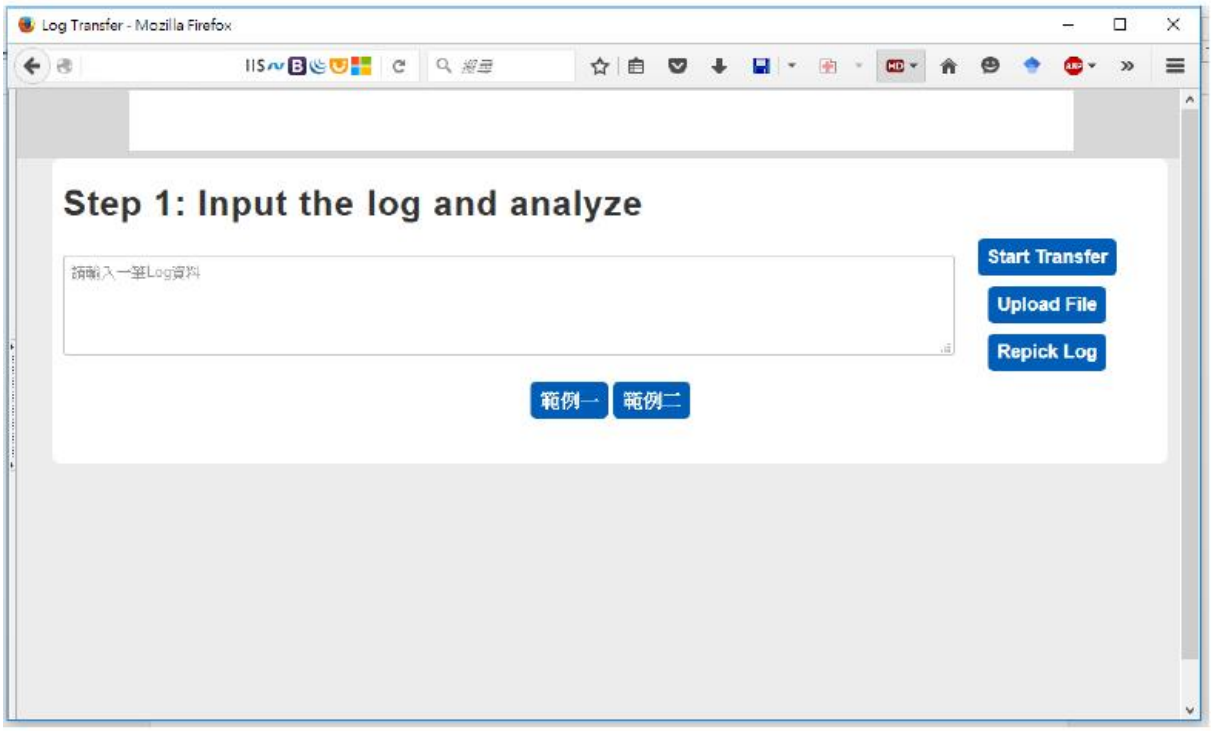
肆、 系統展示

本研究所開發之智慧型 Parser 為一正式上線的系統，而本節將展示該系統如何支援 SOC 作為紀錄檔整合的基礎建設。本節將以個案探討的方式說明本系統運作流程。下圖十三為 AscenLink 系統的紀錄檔。



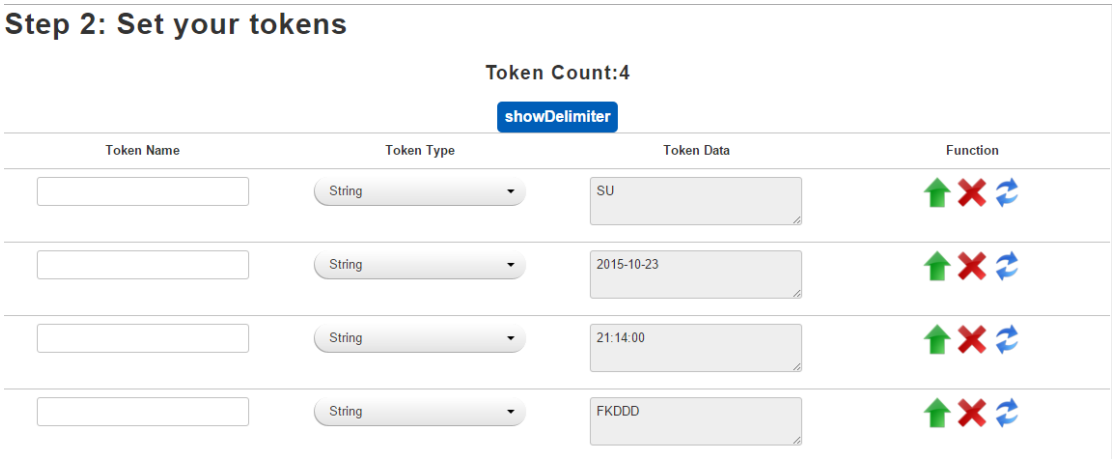
圖十三、AscenLink 系統紀錄檔

本系統為 Web 介面，下圖十四為本系統的首頁。進入首頁後可以選擇手動輸入一筆資料或是上傳整個紀錄檔。如果選擇上傳紀錄檔，本系統將隨機挑選一筆紀錄作為分析用圖。



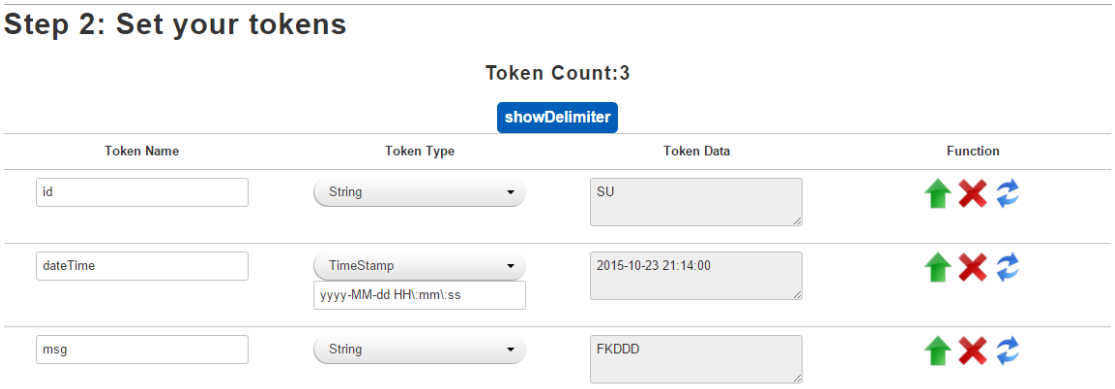
圖十四、系統首頁

本系統可以自動讀取紀錄檔產生相對應 Parser 的正規表示式，並根據常見的紀錄檔如":<>{}'[];,等特殊的字元自動切割成可能的 token，而本研究也將紀錄檔中常見的格式如 IP 位址等作為判斷解析紀錄檔的依據。透過圖形化的介面，讓使用者可以手動輸入一筆紀錄檔或是上傳樣本紀錄檔，當使用者按下開始轉換時，首先會根據紀錄檔拆解 Token。下圖十五為拆解後的畫面：



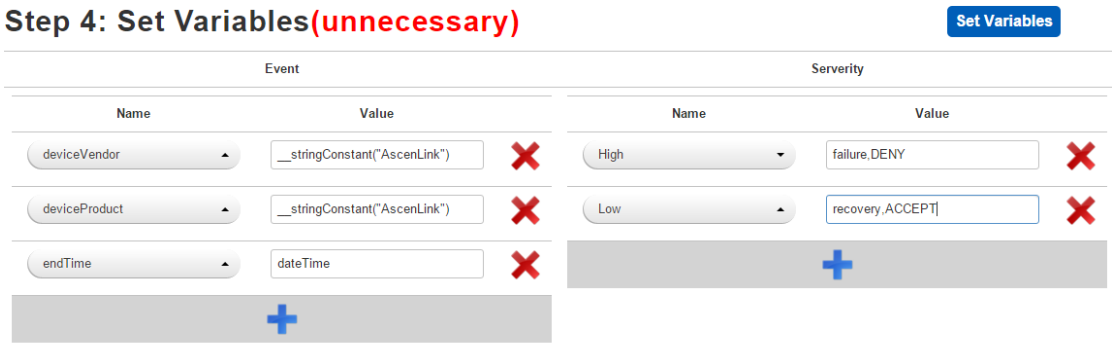
圖十五、紀錄檔拆解 Token 畫面

本系統將根據管理者輸入的紀錄檔自動拆解 Ttoken，畫面最左邊為管理者可以自訂欄位名稱，中間的部分可以選擇該欄位的資料型態而最右邊則處理後欄位所呈現的狀況。如果管理者認為沒有問題，只需按下 Generate Regex 即可以產生相對應的正規表示式與其對應之解析檔案。然而，由於正規化表示法可能無法完全符合需求，而系統操作者也想知道整個紀錄檔剖析的運作方式，因此本研究的系統提供管理者動態調整解析方式的操作功能，在上圖十五的畫面右邊有顯示三個功能按鍵，其中左邊綠色箭頭表示向上合併的功能，舉例來說，假設系統剖析出來的第一個欄位為日期，其格式為 yyyy-mm-dd，而第二個欄位為日期其格式為 hh:mm:ss，系統管理者可以將日期與時間這兩個欄位合併為一個日期時間的欄位，其格式為 yyyy-mm-dd: hh:mm:ss。下圖十六為向上合併的範例。由圖十六所示，將原本時間與日期合併為一個欄位並且將其格式定為”yyyy-MM-dd HH:mm:ss”。而在此步驟可以將每一個 token 命名以及定義資料型態。本範例的三個 token 名稱分別為 id，dateTime 以及 msg。



圖十六、向上合併範例

本系統中間紅色的刪除表示此欄位不需要處理，解析器將自動忽略原始日誌這部分的紀錄以只選取重要的欄位。而右邊藍色箭頭則是將此欄位再進行細部拆解，例如原始欄位包含日期與時間，若此時需要再拆解為兩個欄位，則可以透過點擊藍色箭頭進行處理。而為了作紀錄檔的格式整合，本系統提供變數的功能，透過變數設定的功能可以與 SOC 系統整合。下圖十七為變數功能示意圖。



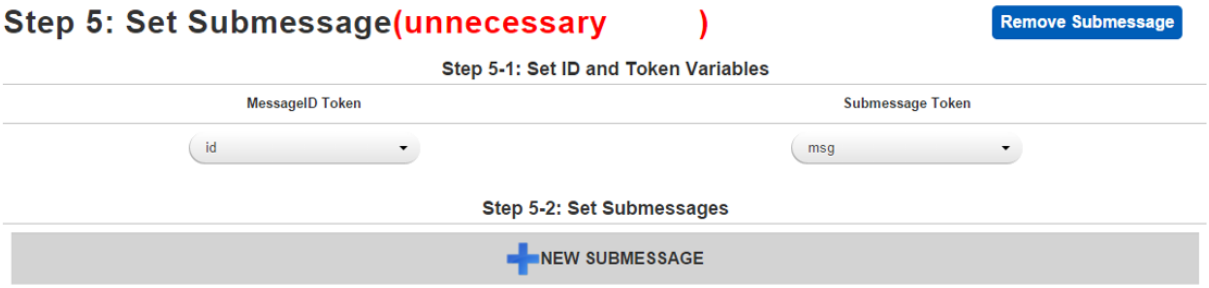
圖十七、變數功能示意圖

本系統提供與 SOC 系統整合的變數設定以及 Severity 設定。此範例設定三個變數，分別為 deviceVdntor、deviceProduct 以及 endTime。分別對應的值為 _StringConstant(“AscenLink”)、_StringConstant(“AscenLink”)以及 dateTime。其中 dateTime 為紀錄檔拆解後的一個 Token。當本系統進行剖析並且將資料匯入到 SOC 時，SOC 系統的 deviceVdntor、deviceProduct 欄位將會填入_StringConstant(“AscenLink”)。而 SOC 系統整合的另一個重點則為 severity。不同的系統對於 severity 用不同的字表達。圖十七的範例定義兩種 severity 的值”high”以及”low”。若 AscenLink 之紀錄檔出現 failure 或是 DENY 則 Parser 將 severity 的值設定為”high”並且輸入到 SOC 系統；若 AscenLink 之紀錄檔出現 recovery 或是 ACCEPT 則 Parser 將 severity 的值設定為”low”並且輸入到 SOC 系統。本系統透過變數以及 severity 的設定整合不同格式的紀錄檔至 SOC。

除此之外，若上傳的紀錄檔存在多階層架構（類似於程式語言中常見的 if-then-else 語法），例如某些特殊狀況需要切割成 3 個欄位傳值，其他時候需要切割成 5 個欄位傳值，本系統也提供該特殊日誌檔處理方式（稱為 sub-message）。舉例而言，圖十三的範例當中。有三筆資料顯示如下

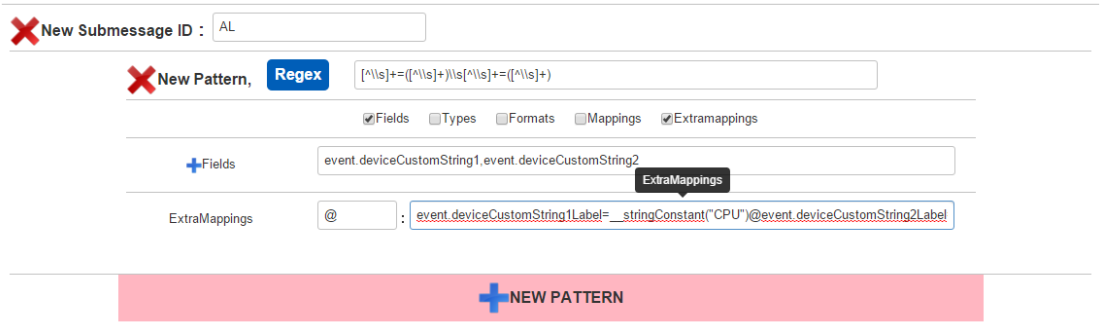
```
SU 2015-10-23 21:14:00 FKDDD
AL 2015-10-23 21:39:01 cpu=7.6% conn=83
FW 2015-10-23 21:13:08 ACTION=ACCEPT PROTO=Unknown(UDP) SRC=10.88.10.5:137 DST=10.88.15.255:137 TOTLEN=78
```

該範例中，同一個紀錄檔有不同的格式以及欄位，其中前三個欄位為共同欄位，但是第三個欄位以後的資料則會根據前面的資料有所不同。以該範例第一筆資料而言，第一個 Token 值為 SU，則後面訊息則顯示 FKDDD；而當第一個 Token 的值為 AL 時，後面的訊息則是顯示 CPU 以及連線數的資訊；而當第一個 Token 的值為 FW 時，後面訊息則顯示防火牆的資訊。sub-message 功能就是用以處理此種複雜得 log 檔。下圖十八為 sub-messgae 介面。



圖十八、Sub-message 主畫面

上圖十八左邊為 MessageID token，以範例而言，就是範例中三條紀錄檔的 AL、SU、FW 資料所屬的 Token。而 Submessage Token 就是對應到的訊息。以程式設計的角度而言 MessageID token 所代表的就是 if 而 Submessage Token 代表的就是 if 後面的 then。因此在 sub-message 功能當中，使用者必須列舉所有 MessageID token 內的所有可能出現的值，並且針對每一個值產生出正規表示式。下圖十九為一 sub-message 設定範例。由圖十九可以得知，系統操作者正在針對當 AscenLink 中的 Log 之 MessageID token 中，若 MessageID token 的值為 AL 時，系統該如何產生正規表示法。



圖十九、sub-message 設定範例

當 Log 檔案的 Token 以及正規表示法設定完畢後，本系統則會產生一個 Parser 設定檔。當 SOC 系統讀取該設定檔後，就可以針對該種 Log 做剖析以及整合的動作。本研究所產生的 Parser 設定檔是以 Arcsight 系統作為目標。然而只要在設檔中加以變化則可

以適用於其他的 SOC 系統。下圖二十為本研究所開發的自動化 Parser 產生器 Parser 系統所產生的設定檔範例，設定檔包含原始紀錄檔格式、正規表示式、token 名稱與 token 型態等，前端事件收集器可利用此解析檔剖析原始設備日誌所需之結構化資訊並導入 SIEM。圖二十一為本系統自動化 Parser 演算法的虛擬碼。

```
# FlexAgent Regex Configuration File
do.unparsed.events=true

#SU 2015-10-23 21:14:00 FKDDD
regex=([^\\s]+)\\s([^\\s]+\\s[^\\s]+)\\s([^\\s]+)

token.count=3

token[0].name=id
token[0].type=String

token[1].name=dateTime
token[1].type=TimeStamp
token[1].format=yyyy-MM-dd HH\:mm\:ss

token[2].name=msg
token[2].type=String


event.deviceVendor=__stringConstant("AscenLink")
event.deviceProduct=__stringConstant("AscenLink")
event.endTime=dateTime

severity.map.high.if.deviceSeverity=failure,DENY
severity.map.low.if.deviceSeverity=recovery,ACCEPT

submessage.messageid.token=id
submessage.token=msg
submessage.count=2

submessage[0].messageid=SU
submessage[0].pattern.count=1
#FKDDD
submessage[0].pattern[0].regex=([^\\s]+)
submessage[0].pattern[0].fields=event.name

submessage[1].messageid=AL
submessage[1].pattern.count=1
#cpu=7.6% conn=83
submessage[1].pattern[0].regex=[^\\s]+=([^\\s]+)\\s[^\\s]+=([^\\s]+)
submessage[1].pattern[0].fields=event.deviceCustomString1,event.deviceCustomString2
submessage[1].pattern[0].extramappings.delimiter=@
submessage[1].pattern[0].extramappings=event.deviceCustomString1Label=__stringConstant("CPU")@event.deviceCustomString2Label=__st
```

圖二十、Parser 系統所產生的設定檔範例

```
DEFINE parenthese as " ' ( ) [ ] { } < >
String FUNCTION RegexGenerate(TokenData[])
    SET Regex TO empty string
    SET Delimiter TO empty string
    FOR i = 0 TO TokenData.count INCREMENT 1
        IF there is a complete parenthese column THEN
            IF TokenData[i + 1] is message THEN
                IF Delimiter is not empty AND doesn't contains TokenData[i] THEN
                    IF TokenData[i].Information is " OR ' THEN
                        ADD [^TokenData[i].Information]+ TO Regex
                    ELSE
                        ADD [^\TokenData`i].Information]+ TO Regex
                    END IF
                ELSE
                    ADD DelimiterRegexGenerate(Delimiter) TO Regex
                END IF
                SET Delimiter TO empty string
                IF TokenData[i + 1].Information contains TokenData[i + 2].Information THEN
                    IF TokenData[i + 1].Information contains whitespace character THEN
                        IF TokenData[i].Information is " OR ' THEN
                            ADD TokenData[i].Information(.*)TokenData[i + 2].Information TO Regex
                        ELSE
                            ADD \TokenData[i].Information(.*)\TokenData[i + 2].Information TO Regex
```

```
                        END IF
                    ELSE
                        IF TokenData[i].Information is '' OR ' THEN
                            ADD TokenData[i].Information([^\s]+)TokenData[i + 2].Information TO Regex
                        ELSE
                            ADD \TokenData[i].Information([^\s]+)\TokenData[i + 2].Information TO Regex
                        END IF
                    END IF
                ELSE
                    IF TokenData[i].Information is '' OR ' THEN
                        ADD TokenData[i].Information([^TokenData[i + 2].Information]+)TokenData[i + 2].Information TO Regex
                    ELSE
                        ADD \TokenData[i].Information([^TokenData[i + 2].Information]+)\TokenData[i + 2].Information TO Regex
                    END IF
                END IF
                ADD 2 TO i
                CONTINUE
            END IF
        END IF
        CASE TokenData[i].Type OF
            :
                ADD TokenData[i].Information TO Delimiter
            String:
                ADD DelimiterRegexGenerate(Delimiter) TO Regex
                SET Delimiter TO empty string
                IF TokenData[i].Information contains whitespace character THEN
                    IF TokenData[i].Information matches regular_expression([A-Za-z][^=]+=) THEN
                        ADD ([A-Za-z][^=]+=) TO Regex
                    ELSE IF TokenData[i].Information matches regular_expression([A-Za-z][^:]+:) THEN
                        ADD ([A-Za-z][^:]+:) TO Regex
                    ELSE
                        ADD (.*) TO Regex
                    END IF
                ELSE
                    ADD ([^\s]+) TO Regex
                END IF
            TimeStamp:
                ADD DelimiterRegexGenerate(Delimiter) TO Regex
                SET Delimiter TO empty string
                IF TokenData[i].Information contains whitespace character THEN
                    IF TokenData[i].Information matches regular_expression([^\s]+\s[^\s]+) THEN
                        ADD ([^\s]+\s[^\s]+) TO Regex
                    ELSE IF TokenData[i].Information matches regular_expression([^\s]+\s[^\s]+\s[^\s]+) THEN
                        ADD ([^\s]+\s[^\s]+\s[^\s]+) TO Regex
                    ELSE
                        ADD (.*) TO Regex
                    END IF
                ELSE
                    ADD ([^\s]+) TO Regex
                END IF
            Date, Time, IPAddress, IPv6Address, Integer, Long, MacAddress, RegexToken:
                ADD DelimiterRegexGenerate(Delimiter) TO Regex
                SET Delimiter TO empty string
                IF TokenData[i].Information contains whitespace character THEN
                    ADD (.*) TO Regex
                ELSE
                    ADD ([^\s]+) TO Regex
                END IF
        END CASE
    END FOR
    IF Delimiter is not empty AND the end OF Regex is not .* THEN
        ADD .* TO Regex
    END IF
    REPLACE \ OF Regex TO \\
    REPLACE mutiple \r\n OF Regex TO single [\\s]+
    RETURN Regex
END FUNCTION
```

圖二十一、正規表示式產生器虛擬碼

透過本研究所設計出的智慧型 Parser 產生器，可以整合異質資料檔至 SOC 系統。不但可以節省網管者的時間，並且可以產生品質良好且一致的 Parser。而更重要的是資料的良好整合有助益網路攻擊行為的發現。

伍、結論

SOC 的建置為組織偵測 APT 以及目標式攻擊的解決方案，不少研究利用人工智慧、機器學習或特殊演算法以偵測 APT 以及目標式攻擊，然而這些演算法必須建構於良好的資料內容品質以及運算效率基礎上。本論文以解決資訊安全維運中心實務問題的角度開發出一套智慧型的剖析器，協助管理人員可以快速地完成剖析系統所需要的正規表示式。本論文所開發的系統不但可以節省組織開發解析器所需要的成本，並可以保證解析成果的品質一致性與提高處理效能。而當面臨如 APT 或目標式攻擊此類需要長時間且大量的資料以進行分析時，傳統的關聯式資料庫無法處理如此巨量資料，因此本研究利用雲端平台技術以解決此問題，而實驗證明本研究所設計的系統大幅提升解決巨量紀錄檔分析效率之效率問題。透過本研究所開發的系統即可以作為 SIEM 資料收集以及正規化的基礎，再透過雲端平台以運用偵測演算法將對於提升分析效率上有顯著貢獻。

[誌謝]
本研究由受安碁資訊『自動化解析器產生系統』計畫以及科計部計畫 MOST 105-2221-E-034 -011 -MY2 補助支持，特此誌謝。

參考文獻

[1] D. Alperovitch, “Revealed: Operation Shady RAT”, McAfee, 2011, (Available at:http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf*)*.

[2] K. Annervaz, V. Kaulgud, J. Misra, S. Sengupta, G. Titus and A. Munshi, “Code clustering workbench,” *2013 IEEE 13th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pp. 31-36, 2013.

[3] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian and J. Nazario, “Automated classification and analysis of internet malware,” *Recent advances in intrusion detection*, pp. 178-197, 2007.

[4] P. He, J. Zhu, S. He, J. Li and M. R. Lyu, “An evaluation study on log parsing and its use in log mining,” *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*,2016.

[5] X. D. Hoang, J. Hu and P. Bertok, “A program-based anomaly intrusion detection scheme using multiple detection engines and fuzzy inference,” *Journal of Network and Computer Applications*, vol. 32, pp.1219-1228, 2009.

[6] Infovision Institute, “Service Offerings,” (available at: http://www.infovision.

com/services/technology-solutions/big-data-analytics/service-offerings/).
[7] Ponemon Institute, "2013 Cost of Cyber Crime Study: United States" Hewlett-Packard Enterprise Security Inc, Oct, 2013. (available at http://www.hpenterprisesecurity.com/ponemon-2013-cost-of-cyber-crime-study-reports).
[8] S. Ramadass, "Malware detection based on evolving clustering method for classification," *Scientific Research and Essays*, vol. 7, pp. 2031-2036, 2012.
[9] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The hadoop distributed file system," *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1-10, 2010.
[10] T. White, Hadoop: The definitive guide: "O'Reilly Media, Inc.", 2012.
[11] Wildlist, (available at *http://www.wildlist.org/WildList/t_archive.htm*), 2013.
[12] J. Zhang, Y. Guan, X. Jiang, H. Duan and J. Wu, "AMCAS: An automatic malicious code analysis system," *2008 The Ninth International Conference on Web-Age Information Management*, pp. 501-507, 2008.
[13] 蘇文彬，趨勢：APT 攻擊從政府單位轉向企業組織，中小企業也受害。趨勢科技，http://www.ithome.com.tw/news/95599, 2015/05/22.

[作者簡介]

黃瓊瑩（Huang, C.Y.）C.Y.Huang@acer.com：台大 EMBA 資訊管理碩士，英國曼徹斯特大學資訊工程碩士，現任宏碁電子化資訊管理中心（Acer eDC）資安維運處處長。專長為密碼學、SOC 營運與管理，資安事件處理與應變。
孫明功（Morgan Sun）Morgan.Sun@acer.com：國立成功大學電腦與通訊工程博士，現任宏碁電子化資訊管理中心（Acer eDC）經理。專長為密碼學、資訊安全，SIEM / SOC 建置與顧問服務。
陳嘉玫:馬里蘭大學計算機科學博士，現任國立中山大學資管系教授。研究興趣包含行動裝置安全、雲端入侵偵測系統以及惡意程式偵測與分析。
龍志雄:現任國立中山大學碩士生，研究興趣包含紀錄檔分析以及惡意程式偵測與分析。
賴谷鑫:國立中山大學資管系博士，現任臺灣警察專科學校科技偵查科助理教授。研究興趣包含惡意軟體分類、IPv6 安全。