TWAREN 骨幹異常使用即時偵測與告警系統

梁明章
國家高速網路與計算中心
liangmc@narlabs.org.tw

摘要

本文將說明 TWAREN NOC 維運團隊利用研究網路骨幹 NetFlow 開發的即時異常使用偵測系統，以 ElasticSearch Cluster 即時儲存 NetFlow 資料建立全文檢索，以 C 語言開發高速的彙整、統計、排序查詢系統作為前端輔助系統，結合兩者開發即時異常使用與攻擊偵測暨告警系統，即時偵測大規模大範圍掃瞄或攻擊、SYN flooding 攻擊、DDoS 攻擊、高流量使用者，說明實作方法，可自動告警並通報相關資訊以供 NOC 或各校網管人員快速反應處理，如攻擊入口過濾、特徵說明等，並留存偵測結果作為進一步研究的資料來源。

關鍵詞：TWAREN、NetFlow、ELK、異常偵測、資安、DDoS。

1. 前言

近年來由於資安議題越趨重要，TWAREN[1] NOC 也越來越重視網路異常使用與攻擊的議題，然而我們沒有偵測骨幹過境封包內容的權力，於是我們能偵測的資料來源只剩下流量數據及 Netflow，儘管我們已經利用 Netflow 開發了各種日報、月報等使用分析，但這些都無法提供即時的異常或攻擊偵測，因此線上即時的 Netflow 彙整、排序、異常偵測成為我們近年來努力的目標之一，我們利用免費的 ElasticSearch[2]建置了一組大資料處理平台，讓 netflow 資料源源不斷地匯入 ElasticSearch 平台中，可以快速地查詢資料，然而我們發現，由於 ElasticSearch 在工作處理過程必須 All-in-memory 不使用硬碟暫存中間過渡資料的特性，在處理多重彙整統計與排序時需求大量的 JVM HEAP 空間放置中間過渡陣列(Field-Data-Array 與 Request-Data-Array)，由於 ElasticSearch 平台面對每秒三四萬筆的 netflow 資料匯入就已經承受很大的 JVM 記憶體壓力，一旦加上大量的過渡陣列需求，就會爆發大量的 GC (Garbage Collection)，嚴重時會引發 JVM 的 Old-Generation-GC，此時會造成工作凍結，導致 Data-Node 無法回應 Master-Node 的 Alive-Check，一一被踢出 Cluster，最終形成崩潰，這樣的問題已發生多次，最正統的解決方法當然是添購很多記憶體來打造更多的 Data-Node (因為 JVM 建議 HEAP 不要超過 32GB 以免效率大跌，所以只能增加 JVM 個數) 來避免 GC，然而這樣的投資報酬率令人難以忍受，所以我們決定將會造成大量 GC 的運算不再以 ElasticSearch 來處理，保留 ElasticSearch 平台作為詳細 Netflow 資料的查詢與驗證核心，開發一些前端程式來處理即時的彙整統計與排序以及異常偵測，我們以 C 語言開發出能即時處理 Netflow 的 IP 與 Link 用量彙整並儲存必須資料至硬碟的程式，處理速度比平時 Netflow 產生的速度快很多，這樣才能有餘力應付網路大量攻擊時暴增的 Flow 數量，並且能在幾秒內回應查詢且排序，查詢消耗的運算代價也很低，能應付頻繁的查詢，適合作為週期性異常使用自動偵測的資料來源基礎，這個系統的運算原理與方法我們已經投稿在 TANet2015，於「TWAREN Netflow 線上即時查詢與 ELK 之實作」[3]一文中作了詳細說明，在本文中不再詳述。

以這個高速系統為基礎，我們可以頻繁多次的查詢排序也不會擔心如 ElasticSearch 那般崩潰，雖然只能提供十幾種特定查詢與排序，但是對於初步的異常偵測而言已經足夠，利用這個系統我們可以查詢 IP 的 Flow 連結數量、Packet 數量、Bytes 傳輸量、TCP 數量、UDP 數量等方面的 Top-N，進行交叉判斷，評估可能的異常嫌疑者 IP，告警給 NOC 作進一步處理，後文將說明我們目前已經開發的一些異常判斷方法。

我們一開始開發此異常即時偵測是以能夠自動告警給 NOC 為目標來設計，為了避免告警太多，我們針對各項判斷數值的臨界值訂得相當高，僅針對「極大規模」才告警，然而我們也有發現，當我們降低臨界值時，嫌疑者就會增加很多，但是搭配 ElasticSearch 進行詳細的 Netflow 內容抽查的結果，發現這些嫌疑者的行為的確都是異常，換言之，在我們告警範圍之外，依然充斥著很多異常行為，只是規模沒有超過我們訂的臨界值而已。我們進一步思考，雖然降低臨界值來抓取大量異常者來進行告警並不實際 (NOC 會很煩，被通報的學校會更煩)，然而我們是否可以將這些不在告警範圍內的異常者也記錄下來呢？雖然有很多學者在研究將 Netflow 大資料匯入大資料平台(如 Hadoop)作運算，可以抓出許多異常行為，然而每次運算都要花相當長久的時間，光是大量 netflow 資料進出 HDFS 就要花費許多時間，相較於 netflow 產生的速度，這種方式很難追上新資料作全面的探測，於是我們就想到，如果我們不追求全部抓到，只想抓到比較大的，其實我們可以在前述的異常偵測時，順便作幾個階段的過濾，存下一些

有用的嫌疑者資訊，然後以半天或一天的規律，利用這些資訊配合 ElasticSearch 系統作綜合細部的交叉查詢，嘗試組出異常者的網絡關聯，只要不涉及大量的彙整統計排序，ElasticSearch 應付查詢綽綽有餘，使用自己開發的程式也可以，目前 ElasticSearch 官方也有與 Hadoop 溝通的模組，也能連結 Hadoop 與 ElasticSearch，只要能夠避免再塞一份資料進 HDFS 的時間與空間，就能用比較適當的代價來進行運算，我們將在後文做更詳細的規劃說明。

2. 異常使用偵測

前文提到的快速統計排序引擎會即時將新進的 NetFlow 彙整統計，建立 IP-Usage 與 Flow-Link-Usage 兩個主要的列表，兩個列表都是固定欄位長度的資料表，IP-Usage 的欄位定義如下表 1 所示，Flow-Link-Usage 的欄位定義如下表 2 所示，快速彙整引擎在即時處理 NetFlow 時也同時分析每個 NetFlow 的 Protocol 欄位以及 TCP-Flags (TCP Control Bits) 欄位，並且將分析結果也統計到 Usage 中，當然，TCP-Flags 只有 Protocol 是 TCP 的時候才有用，儲存這些資料是為了後續將提到的異常偵測，下表中 Flag 變數統計的是 TCP-Flags 中 SYN 與 FIN 旗標的數量，因為在正常情況下，TCP 連線在三向握手時會使用 SYN 旗標，結束連線時使用到 FIN 旗標，如果一條 NetFlow 中包含了一個完整的 TCP Session，那 NetFlow 中的 TCP-Flags 欄位值應同時具有 SYN 及 FIN，如果因為 Session 太長導致被分開成多條 NetFlow，那麼在開頭與最後的 NetFlow 中也應該各自有 SYN 及 FIN，換言之，正常使用下 SYN 與 FIN 的統計值不應差異過大，如果 FIN 遠少於 SYN，表示非正常連線的可能性極高。

表 1 IP Usage 欄位定義

型別	變數名	說明
Uint32	IP	使用者 IP
Uint64	ByteIn	流向此 IP 的 Byte 總數
Uint64	ByteOut	此 IP 輸出的 Byte 總數
Uint32	FlowIn	流向此 IP 的 Flow 總數
Uint32	FlowOut	此 IP 輸出的 Flow 總數
Uint32	PktIn	流向此 IP 的封包總數
Uint32	PktOut	此 IP 輸出的封包總數
Uint32	Tcp	與此 IP 有關的 Flow 中 Protocol 為 TCP 的 Flow 總數
Uint32	Udp	與此 IP 有關的 Flow 中 Protocol 為 UDP 的 Flow 總數
Uint32	Icmp	與此 IP 有關的 Flow 中 Protocol 為 ICMP 的 Flow 總數
Uint16	Syn	與此 IP 有關的 Flow 中 TCP Control Bits 裡出現 SYN 旗標的 Flow 總數
Uint16	Fin	與此 IP 有關的 Flow 中 TCP Control Flag 裡出現 FIN 旗標的 Flow 總數

表 2 Flow-Link Usage 欄位定義

型別	變數名	說明
Uint32	IP1	Flow 連結的一端 IP
Uint32	IP2	Flow 連結的另一端 IP
Uint64	Byte12	IP1流向 IP2的 Byte 總數
Uint64	Byte21	IP2流向 IP1的 Byte 總數
Uint32	Flow12	IP1流向 IP2的 Flow 總數
Uint32	Flow21	IP2流向 IP1的 Flow 總數
Uint32	Pkt12	IP1流向 IP2的封包總數
Uint32	Pkt21	IP2流向 IP1的封包總數
Uint32	Tcp	此連結關係中 Protocol 為 TCP 的 Flow 總數
Uint32	Udp	此連結關係中 Protocol 為 UDP 的 Flow 總數
Uint32	Icmp	此連結關係中 Protocol 為 ICMP 的 Flow 總數
Uint16	Syn	此連結關係中 TCP Control Bits 裡出現 SYN 旗標的 Flow 總數
Uint16	Fin	此連結關係中 TCP Control Bits 裡出現 FIN 旗標的 Flow 總數

在平日無大量攻擊發生時，每五分鐘統計結果，上述的 IP-Usage 列表約有數十萬至百萬筆，亦即此五分鐘內有這些 IP 上線經過 TWAREN 骨幹，而這些 IP 彼此間形成的 NetFlow Link 關聯 (Flow-Link-Usage) 約有數百萬筆，高速排序引擎可以針對 Usage 欄位中的 Bytes、Flows、Packets 各自區分 In、Out、In+Out 共九項複數條件進行 Top-N 排序，排序本身運算很快，時間主要消耗在從硬碟讀取列表檔，第一次需要一兩秒，之後連續查詢僅需不到一秒 (因為列表檔已被緩衝於記憶體中)。由於引擎由 C 語言寫成，自己支配記憶體，不會像 ELK 那般受到 JVM 的32GB HEAP 瓶頸以及 Garbage Collection 的延遲，可以應付連續頻繁的查詢，因此我們就以這個高速引擎作為異常偵測機制的核心工具，目前我們設定高速彙整統計引擎每五分鐘作一次結算，因此異常偵測系統也配合每五分鐘觸發一輪偵測動作，下個章節我們會說明幾種異常偵測的構想與設計。

由於我們的 ElasticSearch 平台是即時持續地接收新生的 NetFlow 資料並且儲存所有欄位資料及索引，導致記憶體壓力非常大，考慮到成本問題，我們盡量以 C 語言自行開發一些輔助系統來執行那些可能會導致 ELK 耗用大量記憶體的工作，我們逐漸將一些可以協助異常偵測判斷的欄位也記錄到高速彙整排序引擎，例如 Protocol，依照目前網路的使用習慣來看，其實會出現的 Protocol 以 TCP、UDP、ICMP 為大宗，異常使用與攻擊也多半是這三種，所以我們將這三種統計也納入引擎統計資料內。而 TCP-Flags 也只有6~9種，而我們比較關心的是 SYN flooding 之類的攻擊，因此 TCP-Flags 的統計也納入儲存。

SYN flooding 這類型的攻擊並不需要爆大量，他的量只要能超過目標伺服器的 TCP Backlog queue 能承受的數倍即可造成服務阻斷，俯瞰整個

骨幹的NetFlow，想靠量大Top-N來找出SYN攻擊可能性很低，因此我們在Usage欄位中紀錄了SYN與FIN的數量，SYN攻擊的特色就是他沒有FIN，事實上很多打一彈就跑的TCP攻擊都是沒有FIN的(例如Fake-Source-IP)，所以只要我們清查IP Usage列表中SYN跟FIN數量差異很大且Protocol TCP占比很高的紀錄，就能縮小範圍，當然這範圍內還包括單純傳輸大檔案尚未結束而還沒有FIN的情況，但是可以計算Packets per Flow比值以及Bytes per Packet比值，正常大檔傳輸中的Bytes per Packet比值應該接近1500，很少小於一百的，再經過這層過濾後，嫌疑者的認定就更進一步，已經足可進入最後人工判斷的步驟，不過由於SYN flooding無須量大的特色，單靠NetFlow其實不能確切認定，最終仍須由被攻擊者來確認。

2.1 偵測大規模大範圍掃瞄或攻擊

身為骨幹維運者，我們並不需要抓出每個異常使用者，實際上我們也無法付出如此龐大的運算資源，所以我們首先關心的是會影響到骨幹或連線單位網路的異常使用，換言之，以「量」為偵測要點，由於骨幹資料量太龐大，我們只能分為幾個階段逐步篩檢以降低須處理的資料量，第一階段是將 netflow 依照 IP 做彙整，然後排序出Top-N 以縮小進一步處理範圍，正常情況下每五分鐘約有數百萬至千萬筆 netflow、數十至上百萬個使用者 IP，參見下**圖1**、下**圖2**，利用自行開發的Netflow 彙整排序引擎分別按照每個 IP 的 Flow 總數、Byte 總數、Packet 總數、LinkPeer IP 總數(連線對端的 IP 總數)、SYN 旗標比例(含有 SYN 旗標的 Flow 數量除以 Flow 總數的比值)等五個方面查詢出 Top-100的 IP (TOP-N 的 N 值可在系統設定檔中隨時調整)，共得500筆，每個IP各得一分嫌疑分，這500個 IP 再進行彙整，IP 相同者合為一筆，其嫌疑分也合併，彙整後大約會剩下三百多個相異 IP，進入第二階判斷。第一階段的 netflow 彙整是隨時即時進行的，而每次排序查詢耗時不到一秒，因此資料處理速度高於資料新增速度，使異常偵測系統可以一直處於即時狀態。

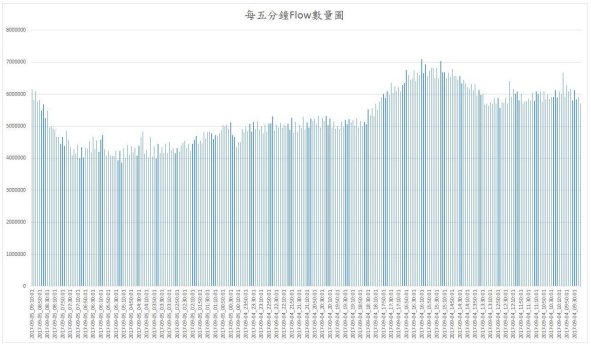

圖 1 每五分鐘 Netflow 數量圖

圖 2 每五分鐘 IP 使用者數量圖

第二階段是綜合多項異常指標數值比對並判斷攻擊者或受害者，進入本階的三百多個 IP 逐個針對 FLOW 數、封包數、BYTES 數這三對雙向數量計算雙向數值的比例，雙向數值差越大表示異常可能性越高，嫌疑分增加，同時推測此 IP 的異常方向，若異常方向是輸出，那麼視為攻擊者，反之則為受害者，嫌疑分累積超過異常指標者才能進入第三階判斷，各項異常指標數值目前由設定檔內指定，未來將研究採用統計學方式用歷史資料推算更加合理的數值，我們目前調整的異常指標大約可讓個位數至十多個 IP 進入第三階判斷。除此之外，SYN 旗標占比 / Flow / Packet / Byte / LinkPeer 數有任一項的絕對數值超過嚴重指標者，也會進入第三階判斷，此嚴重指標是站在網路維運者的角度認為無法忍受的數值，例如某 IP 五分鐘內的平均速率超過800Mbps，由於目前大部分學校依然只有1Gbps 的對外頻寬，一個 IP 就占用800Mbps，就算是正常的使用也很過分，已經擠迫到別人的權益，所以我們認為流量超高應該告警；再一個例子是一個 IP 在五分鐘內就連線超過十萬個相異 IP，這種行為對於出口有防火牆或 NAT 設備的學校來說，極有可能拖累其設備效能甚至造成設備故障，影響其他人權益，因此 IP 連接數超高也被列為嚴重指標。

第三階段開始利用 ElasticSearch 查詢每個嫌疑者 IP 相關的所有 netflow 詳細資料並做多項欄位的彙整及排序以萃取特徵，因此時嫌疑者 IP 僅剩下十幾個，我們的 ElasticSearch Cluster 已可支撐這個查詢量，首先，根據 IP 是攻擊者或受害者進入不同的判斷分支，此時會向 ElasticSearch 查詢該 IP 的攻擊(或被攻擊的)Port 前十名、封包數特徵的前十名、BYTES 數特徵的前十名，如果 LinkPeer IP 數低於一千的話，還會查詢與其相連 IP 的前十名，上述幾個項目的前十名都會判斷其所佔的百分比，百分比越高者特徵越明顯，此階段就是特徵萃取階段。

第四階段會根據多項特徵與數據組合判斷該異常可能是哪種攻擊，組合出文字說明以方便 NOC 人員閱讀，並將前述步驟所蒐集到的資訊形成一篇 JSON 格式的文件，送入 ElasticSearch 等待

進一步研究，內容呈現舉例如下**圖3**：

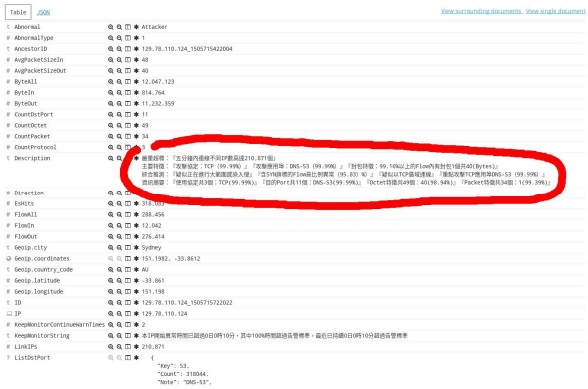

圖 3 異常者資料呈現擷取圖

以上圖為例，五分鐘內連線不同 IP 數高達210,871個(TWNIC 的 DNS 最多才三萬左右)，且99.99%的連線目標都是 TCP port 53，99.16%的netflow 只包含一個封包且 octets 都是40Bytes，40Bytes 只夠放 IP 及 TCP 標頭，換言之幾乎都是TCP 三向握手第一動，沒有第三動之後的封包，綜合上述，這是一個非常典型且特徵明顯的大範圍DNS TCP Port 服務掃描，這樣的規模對骨幹設備來說不算什麼，但是如果該攻擊者 IP 所在單位有防火牆做出口守護的話，幾分鐘內就占用幾十萬個TCP 連線 Session 的資源，又沒有正常結束 TCP Session，如此行為會嚴重傷害防火牆的效能與Session 資源。

DDoS 可分癱瘓目標的服務程式，或是癱瘓目標的網路，後者需要足夠大量才可能癱瘓一個區域的網路，通常在量大的排序中可見端倪，但是前者的規模就不一定，如果目標的服務程式所使用的伺服器能力不大，而且攻擊者也清楚這一點，那麼攻擊者僅需小規模的 DDoS 即可，那就不見得能排入我們所定的 Top-N 內了，在此我們先討論大規模的 DDoS。

大規模 DDoS 若使用大量變動的 Fake-Source-IP，會造成大量的 NetFlow 集中到被害者 IP 上，只要用 Flow In+Out 的 Top-N 排序，再判斷 Flow 的流向是否都是流入集中的 IP 就能偵測到，使用 IP Usage 紀錄的 TCP/UDP/ICMP 三值比例可初步推斷其 Protocol，至於使用的 Port，異常偵測程式可以用被害者 IP 去 ElasticSearch 查詢相關 Flow 的 Port，即可統計出攻擊的 Port (應用服務)。

DDoS 對於被害者與骨幹維運者來說都很難過濾與阻擋，尤其是浮動 Fake-Source-IP 類型的，路由設備 ACL 很難設定，在被害者近端過濾其實跟服務阻斷也沒兩樣，另一種方式是去攻擊進入骨幹的入口作過濾，犧牲較小，但對於 NOC 來說比較麻煩，不過異常偵測程式可以從 ElasticSearch 查出所有相關 NetFlow 經過的 Router-IP 與 Router-Port，進行歸納統計與排序，納入通報資料給 NOC 參考，NOC 可視情況挑量大的入口作 ACL 過濾，協助被害者降低壓力，一般情況下，排序最前的路由器介面應是最靠近受害者或攻擊者的介面，這項資訊也可以幫助 NOC 維運者快速找到著火點，尤其是 Fake-Source-IP 類型的 DDoS 攻擊，無法利用來源 IP 推測攻擊發起處，唯有利用整體骨幹的netflow 資料才能查找來源，由於異常偵測系統能對流經介面做統計排序，因此 NOC 工程師能依此資料考量封鎖點，減輕受害者近端網域的攻擊壓力，若攻擊來源有來自連線單位學校者，還可通知各校網管進行清查或封鎖被利用的 Bot 電腦。

系統每一輪(目前訂為五分鐘)自動偵測總能抓到許多異常，都會產生 JSON 文檔存到ElasticSearch，可做為每小時或每天週期的非即時偵測的資料來源之一。

2.2 告警機制

當異常指標臨界值設定越低，則抓到的異常者越多，我們曾調低到每輪都能抓出數百個異常者，從 ElasticSearch 查詢詳細資料一一觀察，都很明顯是異常行為者，然而作為骨幹網路 NOC，每五分鐘發出幾百個告警實在過度騷擾，因此，紀錄歸紀錄，實際上告警仍應過濾，必須區分嚴重等級，因此我們針對多項異常指標各自設定「嚴重超標」臨界值，主要針對可能會立即危害到連線單位網路服務的狀況，例如連線 IP 數超過十萬個可能會癱瘓效能較差的路由器或 NAT 設備、每秒產生 Flow 超過3000筆也可能會癱瘓學校出口設備、每秒傳輸超過800MBps 可能會吃光某校的1GE頻寬等等幾個嚴重狀況才會立即觸發告警，至於沒有「嚴重超標」的異常 IP 則會先進入系統追蹤名單。

系統每輪異常判斷後會將本輪異常者 IP 跟追蹤名單中的 IP 做比對，IP 已存在名單內則資料更新，若不存在就新增追蹤 IP，若某 IP 在追蹤名單中已連續異常達30分鐘則會觸發第一次告警，此後若異常達到 4 / 12 / 24 小時會再度告警，超過一天後則每日告警一次，只有長時間(目前訂為一日)連續未再度異常者才會被移出追蹤名單，以此機制減少告警數量，以免 NOC 或各校網管疲於奔命，不過付出的代價就是告警觸發的時間會晚於實際案發時間30至40分鐘。

我們將 netflow 資料匯入 ElasticSearch 的即時轉換程式也是自己用 C 語言開發的 (因為 ELK-Stack 的 LogStash 效能追不上骨幹 netflow 新增的速度)，該程式在轉換時會順便查出來源及目的 IP 的GeoIP 資訊，而我們的彙整查詢系統也會在回應結果時從 TWAREN 與 TANet 單位路由資料比對該 IP網段所屬的單位，所以偵測系統可以判斷異常者是否屬於 TWAREN 或 TANet，可據此決定後續的告警分級與通報手段，因此告警信也可以同時寄發給 IP 所屬單位的網管 (目前我們有 TWAREN 連線單位的聯絡地址，若能獲得 TANet 所有學校聯絡

資料也能一併服務)。

下**圖4**為告警信之擷取圖例之一，此例比較特殊，告警原因是「嚴重超標」，其流量超過我們定義的忍耐極限「每秒100000000個 Bytes」，而且異常偵測系統能自動配對出上傳者與下載者，從自動判斷說明來看，應該不是惡意攻擊，只是因為流量太高可能危及連線單位頻寬才告警，在事實發生後十分鐘內 NOC 就能接到告警，若是同步通知連線單位網管的話，該校網管也能快速反應決定是否進行壓制，因此我們的異常偵測系統可以即時為連線單位進行頻寬把關、快速反應打擊流量大戶。

圖4 流量嚴重超標告警信擷取圖

下**圖5**是比較常見的異常告警信，第一個 IP 是相當典型的大範圍 TCP 服務掃瞄，所有 Flow 都只有三向握手第一動封包，目的 Port 分散均勻，看得出是循序變化的，此攻擊是大範圍掃瞄 IP 並探測有開啟哪些服務 Port，以上特徵雖然很典型，但雷同性太高，很難鎖定到單一的 Botnet，較少研究的價值，追蹤機制報告該 IP 被追蹤時間已持續一天，且當中100%時間都是異常的，完全沒有舒緩的跡象。

第二個 IP 199.48.164.35則是利用 UDP，特徵是完全針對 Port 5060 (SIP 網路電話協定用)，每筆 flow 只有一個封包且大小為435 Bytes 上下，而且五分鐘內連線過三萬多個 IP 都是這個封包大小，算是比較特殊的特徵，可能是某種探測或入侵用的惡意程式，如此明顯的特徵，可以做為查詢條件從 ElasticSearch 找出其他類似的同夥，這些攻擊者可能因為規模不夠大沒排入第一階段的 Top-N，但是在這個特徵下將無所遁形，如果把查詢時間範圍拉長，抓出足夠多的同夥，甚至可能從這些同夥的連線關係中重疊萃取出 Botnet 的指揮者。

圖5 流量嚴重超標告警信擷取圖

由於異常者詳細資料內容甚多，且為巢式結構，若用 Plan-Text 模式寄信不易察看，因此信中會附上我們開發的查詢系統 URL，只要點擊 URL 就能連結顯示網頁，下**圖6**即為網頁的擷取圖。

圖6 異常者詳細資料顯示介面擷取圖

如上圖 IP 199.48.164.35 的詳細資料顯示，五分鐘內產生 flow 67402筆，連線相異 IP 卻高達36817個，平均每個 IP 不到兩個封包，不像正常的 SIP 應用，結合前段所述，這個案例特徵就能作為進一步的研究，比較可能鎖定到某個特定的 Botnet。

3. 結論與未來工作

由於我們只有一套 ElasticSearch Cluster，即時收集 TWAREN 骨幹所有 NetFlow，持續不斷的資料湧入、索引建構、分散儲存等等造成 ElasticSearch 平台的 JVM 記憶體時時處於緊張狀態，若同時頻繁使用消耗記憶體較多的彙整統計排序類的工作很容易造成這組 ElasticSearch 平台崩潰，若開啟 ElasticSearch 的保護機制，雖然不會崩潰但卻會造成大一點的工作無法完成。考慮到成本經費問題，我們不傾向一直花錢擴充 Cluster，所以將努力方向轉向如何更加經濟有效的利用現有資源，我們利用 C 語言開發幾種輔助程式以特殊的索

引跟運算來分擔大量可預期的第一階彙整統計排序工作，這些輔助系統可以被頻繁地重複操作，代價小而效能高，在這個基礎上，我們開發了幾種異常偵測的機制，能以三至五分鐘一輪迴，快速偵測幾種異常使用或攻擊，後續階段自動利用 ElasticSearch 查詢細節資訊，納入通報提供 NOC 參考，縮短 NOC 處理的時間，例如大規模 DDoS 時提出路由器介面排行有助於 NOC 快速考慮 ACL 過濾應設定的地點，對連線單位網路有威脅的流量大戶或連線大戶也能快速偵測並告警，可協助連線學校維持網路品質。

我們仍在持續開發更多的異常偵測判斷，同時也嘗試將已經開發的異常偵測更加細化，作更精確的診斷並自動附帶更多更有用的資訊，主要是改善程式使其能交叉判斷更多條件。

在我們的異常判斷中，很多地方都牽涉到臨界值的設定，例如骨幹 NetFlow 總數超過四萬或暴增比例超過兩倍等等，這些臨界值其實只是觀察經驗值，事實上很多臨界值如果調高或調低，都會使浮現的嫌疑者增加，我們曾實驗過調低臨界值並查看每個嫌疑者 NetFlow 詳細內容，排除掉那些白名單的伺服器之後，幾乎都確定是有異常行為的，但因為 NOC 不可能每五分鐘都要處理這種告警，嫌疑者所在的學校網管也不願意，所以我們很鴕鳥的提高了很多臨界值，目前甚至一個月才只會告警幾次而已。

然而我們已經想到，如果我們把降低臨界值時浮現的那些嫌疑者有用的資訊記錄在資料庫而不告警，每五分鐘累積下來，一天或一周執行一次大跨度的綜合判斷，是否能挖掘出一些原本不顯眼的訊息？例如同時間爆發的攻擊者其相關連結的 IP 中是否有不像被攻擊或掃瞄的重複者？不同發作時間但手段類似的攻擊者們是否也有這樣的重複者？這樣的挖掘或許能找到 C&C 也說不定，諸如此類，因為這些紀錄的嫌疑者已經是經過即時偵測系統初步過濾的，相比從一周總量的 NetFlow 大海裡茫然撈針而言，利用這些初步紀錄已經可以省下很多大資料運算的資源，也能少花很多時間。

我們將每一輪偵測到的異常 IP 資料，無論是否已存在於追蹤名單中，都會將新資料寫入 ElasticSearch 中，因為異常者有可能隨著時間改變攻擊手段，因此持續的紀錄是必要的，不過這些後續的紀錄都會標記著最開始被追蹤的那筆紀錄，可輕易地拉出整個關係串列，這樣的串列資料有助於快速分析這些已經被 Botnet 掌控的 Bot 其運作特徵的演變，或許這些演變也能形成特徵用來區分 Botnet 群。

以上所言是我們的未來工作之一，TWAREN NOC 團隊會盡力加強骨幹網路資安的研究，以協助各連線學校的資訊安全防衛，我們先以維運所需的即時偵測告警為目標，目前已有初步成果，而副產品則是已經過初步分析的異常資料，這些不斷累積的異常資料，可以讓我們不需要每次都得從大海般的原始資料中重撈一次，大大節約運算資源需求，讓我們可以不用持續投資購買更多的儲存及運算資源，同樣的資源能做更多事，自然提升了投資報酬率。

除了可對連線單位告警之外，目前我們也在開發異常偵測系統的使用者介面，將來系統可有限開放給連線學校網管做查詢與報表產出，規劃中未來可讓連線單位網管設定該校專有的異常臨界指標或嚴重指標，而不再是全骨幹統一標準，可以更加貼合各校的需求。

參考文獻

[1] TWAREN, TaiWan Advanced Research and Education Network. http://www.twaren.net/.

[2] https://www.elastic.co/.

[3] 梁明章, 陳俊傑, “TWAREN Netflow 線上即時查詢與 ELK 之實作” in 暨南大學, TANET2015 臺灣網際網路研討會, 2015.