骨幹大規模異常行為特徵自動萃取與搜尋 Botnet 之研究

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 利用骨幹 Netflow 大資料研究骨幹大規模異常行為的特徵萃取，然後利用這些特徵從 Netflow 大資料中尋找同類的小規模異常者進行判斷與紀錄，並將這些程序實作為自動化機制，以協助相關學術研究或使學術單位網管可以依此清除受感染者或進行 Botnet 傳輸內容研究，同時也說明一些在節省硬體經費的前提下優化大資料演算的心得跟各校分享。

關鍵詞：Netflow, 大資料, Botnet。

1. 前言

TWAREN[1] NOC 團隊利用骨幹 Netflow 資料來運算即時異常行為偵測告警已大約兩年，我們可以即時偵測到 DDoS 攻擊以及大規模掃描探測或入侵行為，這些運算都需要處理骨幹大資料，因此我們對於大資料運算也累積了一些經驗，可以在本文分享。

骨幹網路 Netflow 資料的產生速度相當可怕，對於大資料平台的衝擊壓力主要呈現在新資料每時每刻都在進入，隨時都在建構索引，又因為資料體量龐大(不光是筆數多，占用 Bytes 也多)，要即時複製到另一座大資料平台進行查詢也不切實際，因此索引建構與即時查詢都壓在同一組大資料平台導致壓力龐大。

TWAREN NOC 團隊使用 ElasticSearch[2]作為大資料平台(後文簡稱為 ES)，使用16台中階伺服器，每台320GB 記憶體，六顆4TB 硬碟，所以每台機器運行六個 ES Data-Node，每個 JVM 的 Heap-Memory 設定為31GB，根據 ES 官方建議，每個 Node 其實需要另外一倍的記憶體作為 Lucene 引擎的 IO 緩衝記憶體，實際上六個 Node 需求360GB 記憶體，所以我們算是超載使用，但是 TWAREN 的 Netflow 資料量使我們不得不如此做。

ES 將一個資料庫稱為一個 INDEX，可以隨意命名，一個 INDEX 拆分成多個 Shard(分片)，原則上一個運算單位(Data-Node)最多收容一個 Shard 是最安全的分配。當 Client-Node 接收到來自使用者的 Search-Request 時，Client-Node 會先分析該查詢涉及到哪些 INDEX，然後查出每個 INDEX 有哪些 Shard，哪個 Shard 目前位於哪個 Data-Node，再來就把 Request 分發到那些 Shard 所在的 Data-Node，每個 Shard 獨立運行一個 Lucene Engine 來計算，然後將結果送回給 Client-Node 進行彙整。

所以一個 INDEX 拆分的 Shard 越多時(為安全考慮，Shard 數量最好別超過 Data-Node 總數)，查詢所需時間越短，需要付出的代價就是佔用磁碟與記憶體空間更多，而且 ES Cluster 維護越多 Shard 會越吃力，Cluster 出意外後的修復時間會更久，所以 INDEX 拆分 Shard 的數量並非越多越好，下一段會進一步說明。

由於 JVM 在 Heap 小於32GB 時會使用 Compressed OOPS 技術來壓縮物件的定址為32位元以節省記憶體，若 Heap 超過32GB 就無法使用此技術，會衍生浪費與問題，所以 ES 官方建議單個 JVM 最好別超過32GB[3]，因此網路上一些達人就根據這個限制測試 Shard 大小與效能關係，有些結論是假如 Heap size 設定32GB 的話，Shard 在10GB 左右比較好，因為 Shard 太小會造成 Cluster 內 Shard 總數過多增加 Cluster 管理負荷(尤其是出問題時修復會更慢，甚至修到崩潰)，若超過10GB 則是會對 JVM 內其他工作的記憶體擠壓太嚴重(例如即時索引建構以及查詢工作建立臨時陣列等等)，因此我們建議對於資料龐大的 ES Index 宜以每日一個 Index 為佳，然後將 Index 內 Primary Data 總大小除以10GB 得到的商值就是適當的 Shard 數，Shard 不需要一成不變，可以隨著資料增長而調整，但是每個 Index 的 Primary Shard 數量不可超過 Data-Node 總數。

在兩三年前我們剛開始用 ES 平台時 TWAREN 每日 Netflow 進入 ES 後大約是1TB 的量，當時每台機器只做4個 Node，一共64個 Node，1TB 分散成64個 Primary Shard 等於每個是15GB，而且新 Netflow 資料持續地匯入建構索引也造成很大的記憶體壓力，導致進行比較多日時區間查詢時因為記憶體搶占更新太頻繁，GC (Garbage Collection) 凍結時間太久而導致 ES Cluster 崩潰，後來我們做了很多優化的工作，也明白了資料量與 Shard 的平衡問題，擴充到96個 Node 後，將每個 Primary Shard 降到11GB 左右，大有改善，不過在100G 骨幹啟用之後，Netflow 產量逐漸升高，目前已達到每日1.4TB，每個 Shard 又提升到15GB 了，又再度走回鋼索上，而我們卻沒有多餘的 Data-Node 再增加 Primary Shard 數量，由於我們不可能一直投入經費去擴充 Node 數，因此我們花了不少心血在研究如何不增加 Node 的情況下降低崩潰的風險，我們的心得主要有下列幾點：

＊每一次的統計彙整不要產生數千萬甚至億等級的 Bucket 數量。

＊將複雜的複合查詢拆開成多階段，階段之間的成果可以導出給自己寫的程式進行適當的篩選、計算成為濃縮的結果，再進行下一階段查詢。

＊根據查詢之間的相依關係調整順序，盡量使前次查詢產生的中途陣列或 IO 緩衝區資料可以被下次查詢引用。

舉些簡單的例子，假設我們想連續查詢一萬個可疑的 IP 在一個月內的行為統計與模式，一般程式的寫法大概會一次查詢一個 IP 的一個月行為

資料然後重複一萬次，但這種作法很容易造成 GC 過度而凍結，但若是改成一次讀入一日的資料，分別查過一萬個 IP，然後再讀下一日的資料，依此類推，使每日的 IO 緩衝資料可被重複使用，省下的記憶體資料更換與 IO 是非常可觀的，尤其像是我們 Netflow 的每個 Shard 高達15GB 的情況，不但降低 ES 崩潰的風險，更省了很多 IO 的時間，相對要付出的代價就是本來可以全靠 ES 查詢就解決的計算，變成要自行撰寫較複雜且多階段的程式來處理中途的資料，不過確實可以用比較少的資源成本來做更多事。

由於我們開發的幾種分散壓力與提升效能的方法頗有成效，我們才能夠進行更進階的異常偵測，目前我們先將研究重點放在 Botnet 的收集，國網中心已有一個資安團隊利用 HoneyPot 及資安設備捕捉惡意軟體分析行為與 C&C，但只能捕捉落入陷阱者或能複製到封包的線路，而我們想以「面」的方向進行，以骨幹 Netflow 做全面性的 Botnet 偵測，雖然精確度不如 HoneyPot「點」的做法，但優點是涵蓋面廣，結果可以互補並互相參考，本文將要說明我們如何利用骨幹 Netflow 大資料來尋找 Botnet 的成員相關的研究。

主要內容

1.1 即時異常偵測系統產生第一階段資料

TWAREN Netflow 異常即時偵測系統會對每個異常事件分析其特徵，每個異常者 IP 都會對 IP-Protocol、TCP Control Flags (如果 Protocol 包含 TCP)、目的 Port、Packets per Netflow、Octets per Netflow 個別統計彙整成陣列並排序，這些陣列都包含在偵測結果的 JSON 文件內，自動送到 ES-Cluster 保存，同時將陣列內容以人類可讀的文字 EMAIL 給 NOC 進行告警，舉例告警信部分內容如下：

※Attacker：203.xxx.xxx.115（xx 中心）
※異常類型：連線對象過多
※異常超標：五分鐘內連線不同 IP 數高達 61,812個
※主要特徵：『協定：TCP（99.98%）』『應用埠：SSH-22（99.94%）』
※資訊摘要：
＊使用協定共3個：TCP(99.98%)
＊目的 Port 共23個：SSH-22(99.94%)
＊Octet 特徵共1054個：180(13.49%)：120(8.43%)：1915(7.64%)：60(7.50%)：244(6.65%)：40(6.49%)：240(5.95%)：1931(4.45%)：1811(2.92%)：220(2.38%)
＊Packet 特徵共34個：3(15.74%)：1(15.06%)：5(13.55%)：14(12.54%)：2(9.51%)：4(7.07%)：12(5.23%)：11(4.51%)：7(4.12%)：13(4.06%)
＊本 IP 開始異常時間已超過0日0時5分，其中100.0%時間超過告警標準，最近已持續0日0時5分超過告警標準

上述例子首先表明觸發告警的主要原因是五分鐘內連線六萬多個 Unique IP，這算是頗有規模的行為，因此進行異常記錄。程式自動判斷幾個重要陣列的凝聚程度，譬如本例子的 IP 協定雖然使用了3種，但是 TCP 卻占了其中99.98%，而目標埠雖然記錄到23種，可是 Port-22 (SSH)卻占了99.94%，可以說此異常行為完全針對 SSH，而 Packet 與 Octet 種類較多，但能排得上號的特徵也都是10種左右，百分比差異不大，多樣的 Packet 與 Octet 組合顯示並非僅有三向握手行為，不是單純掃描 SSH 服務存在與否，而是真的包含有後續內容，可能包含了惡意碼或是密碼猜測，對於 NOC 來說，網路上充斥著 TCP 三向握手服務掃描已經是生活日常了(特徵就是一條 netflow 只包含一個封包而 Octet 是40Bytes，等於標準 IP+TCP 標頭之和)，反而是帶著惡意碼或探測碼的大規模行為比較少見，彷彿是故意高調行事的，應該是駭客準備的棄子，不過他 Netflow 中包含的 Packet&Octet 特徵卻是可以記錄學習的，駭客發布給 Bot 探測入侵任務時或許會注意到同一單位內的 Bot 不要做同樣的行為以規避網管的歸納運算，但我們是 NOC，Netflow 包含整個骨幹，駭客要細緻到全骨幹內每個 Bot 都使用不同的惡意碼組合恐怕很困難，畢竟惡意碼不是大白菜一樣隨手摘都有的，如此例複雜的特徵如果在全骨幹大資料搜索中還能找到類似特徵的 IP，那麼其屬於相同 Botnet 家族的可能性非常的高。

下面所舉的例子是即時偵測最常發現的典型：

※Attacker：140.1xx.26.68（xx 區網中心）
※本 IP 開始異常時間已超過0日0時25分，其中100.0%時間超過告警標準，最近已持續0日0時30分超過告警標準
※嚴重超標：『五分鐘內連線不同 IP 數高達 606,723個』
※主要特徵：『協定：TCP（100.00%）』『應用埠：2375（99.34%）』『封包數與大小：98.54% 以上的 Flow 內有封包1個共40(Bytes)』『SYN 比例超過95%』
※異常類型：疑似大範圍掃瞄或入侵
※資訊摘要：五分鐘內『連線對象606723個』『使用協定共2個：TCP(100.00%)』『目的 Port 共5個：2375(99.34%)』『Octet 特徵共311個：40(98.49%)』『Packet 特徵共26個：1(98.60%)』

上面這個例子就是很單純的探測目標是否有開 Port 2375應用，從 SpeedGuide[4]查詢得知主要應用是 Docker REST API，因為98%以上的 Netflow 只包含1個封包40Bytes，表示攻擊者對每個目標只發出一個三向握手第一動 SYN 封包就立刻轉移下個目標，如果後續有接到目標回傳的三向握手第二動封包，表示該目標有開啟服務且沒有受防火牆阻隔，可以列入進一步入侵的名單，

分析即時偵測系統產生的報告，可以讓我們推測出不少異常行為模式，這些都會累積進我們的學習資料庫，然而我們並沒有足夠的運算力去為每個 IP 都做如上例般細緻入微的分析，基於 NOC 的角色，以維持網路品質、快速反應為要點，面對持續產生的海量 Netflow，因此我們針對多項規模指標綜合排序，每五分鐘挑出前五百名疑似異常者進行上例的細緻分析，萃取異常特徵，根據多項規則評估異常分數，異常分數高過臨界值者形成 JSON 報告文件記錄到 ES-Cluster 中，事實上我們曾人工檢視過前五百名的細部資料，其實都可以確認是異常行為，只是擔心全部都做細緻分析的運算負載會拖垮已經壓力很大的 ES-Cluster 才故意調高臨界值以減少最後的異常者數量。

而這些骨幹即時異常偵測所產生的報告 JSON 文件，就是我們預備用來做進一步分析的材料，因為我們只有一組 ES-Cluster 平台用來即時接收、儲存 Netflow 資料，所以查詢運算也只能在這一組平台進行，為了降低平台崩潰的風險，我們特別設計將長時區間超大資料的運算工作拆分成多個階段，前述 JSON 文件就是第一階段資料，藉由即時偵測系統的運算結果順便完成，而我們下一階段的目標就是萃取特徵以搜尋 Botnet 成員了。

1.2 異常特徵萃取方法

為了研究萃取方法，我們做過多項日統計、月統計的分析，也觀察一些資料的長時間趨勢，例如下**圖 1**顯示了2018一月份以來的每月異常告警事件數，也等於前述異常報告 JSON 文件數，從圖中可以發現規模大到能被骨幹察覺的異常大部分時間都以掃描、探測、攻擊特定服務佔大部分(也就是目的 Port 的彙整陣列能凸顯出高佔比的前幾名)，至於沒有特定針對的服務，而是循序掃遍每個 Port 用來查探有開啟的服務埠這種基本掃描行為常態居於第二名。至於被 DDoS 的異常事件為何遠低於攻擊別人的事件，主要是因為 TWAREN 領域內的使用者被 DDoS 的機會真的不多，因為攻擊來源是來自商業網路的通常會從 TANet 對外頻寬進入到 TANet，會過境 TWAREN 骨幹對連線單位的攻擊通常來自歐美學研網路，學研網路一般是比商業網路乾淨的，畢竟有學校網管會關心自家校園的異常使用，因此我們的統計結果是抓到攻擊者的事件數遠高於被 DDoS 的受害事件數。

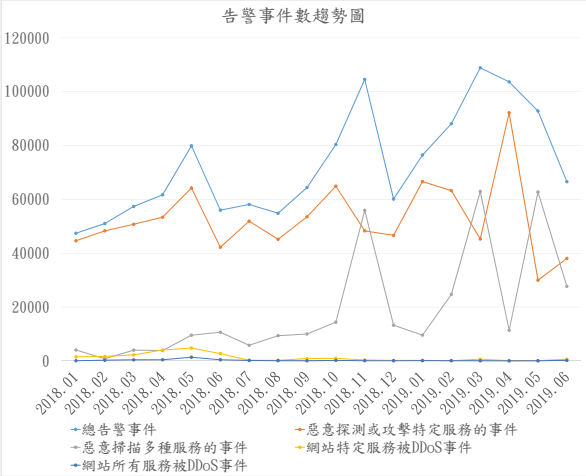

圖 1 每月異常告警事件數趨勢圖

從上圖可以確認特定應用埠不但是個很重要的特徵，而且還是占比最大的異常類型，因此以應用埠為基礎來搜索大規模異常者的行為，將 Netflow 中的 Packet 與 Octet 欄位具備集中性特徵的萃取出來，初步先將集中度超過50%的組合視為主要特徵，例如某個異常者他的 Netflow 資料有超過一半以上都是攻擊 Port 22且包含3個 Packet 與 244Bytes 的 Octet，這就視為主要特徵，以這樣的概念撰寫程式進行查詢及運算，我們更進一步統計分析，結果舉例如下，第一個欄位 xxx-yy-zz 表示 Port-xxx 有 yy 個 Packets 及 zz 個 Bytes 的特徵，而攻擊者數量是指規模大到被偵測系統告警的 Unique IP 攻擊者有幾個：

```
Port 22 | 發現特徵12種
1#22-1-52：8個攻擊者：
    103.207.36.130(@Hanoi[VN])……
2#22-4-240：5個攻擊者：
    185.232.67.11(@Timisoara[RO])……
    …
3#22-1-48：76個攻擊者：
    139.220.192.57(@Bei-
    jing[CN])……
……
……
Port 1433 | 發現特徵9種
1#1433-1-40：55個攻擊者：
    125.64.94.220(@Chengdu[CN])……
……
5#1433-3-152：2個攻擊者：
    222.74.48.230(@Baotou[CN])
    218.4.133.202(@Nan-
    jing[CN])……
……
……
```

上例的 Port-22是 SSH 應用埠，而 Port-1433是 MsSQL 應用埠，都是 TCP 協定，因此去掉第二層(Ethernet 層)標頭之後最小的 Octet 值是40Bytes，因此1433-1-40表示每筆 Netflow 僅僅只有三向握手第一動封包，後面就沒了，非常典型的純掃描應用是否有開或是參與 DDoS SYN 攻擊，並沒有完成三向握手，而22-1-52跟22-1-48可能是動用了 TCP 的 Option 選用功能，因為 Option 增加的 Bytes 會以4Bytes 為單位，52跟48都符合這個條件，不同攻擊者動用相同的 Option 雖然有些特殊巧合，但不能說是強特徵，也都沒完成三向握手，這類型的攻擊必須結合判斷連線目標 IP 是發散或集中，發散表示在廣域掃描服務，為了得到對方回應的三向握手第二動封包，來源 IP 並須是真實的(不過若是駭客能在路徑上攔截封包那麼 IP 就不一定要真實，只要能經過攔截點即可)。如果連線目標 IP 是集中的，那就可能是在參與 DDoS，這種情況就不需要使用真實的來源 IP，參考價值低，處理難度高，本文先不探討這一型的研究。

然而22-4-240與1433-3-152就不同了，以封包數來看，應是完成了三向握手，而且可能還帶了點應用層協定、指令或惡意有問題的格式，可能為了獲取更多資訊(例如伺服程式版本)，同時也表示來源 IP 的真實程度比較高(為了接到想要的資訊)，而且，這樣的組合數字較有特色，當不同的攻擊者恰好用了相同的特徵組合，那麼他們屬於同一個 Botnet Master 指揮或是利用了相同的偵測入侵手段的機率是比較高的，可以先劃分為同一個群組。

因此，當封包數與 Octet 數越高且平均封包大小(亦即 Octet 除以 Packets)超過標準表頭越多(例如 TCP 標準表頭40Bytes)，其特徵性越強，後續研究獲得成果的可能越高，而且在我們 ES 平台運算資源緊繃的情況下，程式自動按特徵性強弱排序後再依序處理，可以讓我們方便掌握處理迴圈數的安全界限以降低崩潰風險，而排序的依據還可以加入異常者數量因素，參與同一種特徵的異常者如果越多，那表示該 Botnet 家族可能更大，所以能拿出更多棄子來揮霍。

1.3 以特徵尋找小規模異常者

我們曾經對國網中心的日常使用做過分析，因為國網中心對外連線一定會經過 TWAREN 骨幹，所以我們可以獲得國網中心對外全部的 Netflow，以上班日全天資料統計結果，可以發現一般人使用的電腦 IP 一天累積下來對外連線的目標 IP 數從幾十到一千多個而已，超過就不正常了，達到數千數萬的通常是伺服器，而伺服器一般是被連線的，這點從 Netflow 的 Source-Port 可以區分，個人連線的 Source-Port 大多是高位區(Port 32768~65535)，而伺服器的 Source-Port 通常低於一萬且多屬於 Well-Known service port，這是我們程式自動判斷異常者是伺服器或一般人的重要規則。當然，如果一個伺服器有一大堆使用高位 Source-Port 的對外連線，會讓我們的程式誤將它當成一般使用者而使用低臨界值觸發告警，不過他的系統管理者也的確該檢查伺服器有如此多的對外主動連線是否正常。

就如上例，我們以1433-3-152做過濾條件，亦即 destinationIPAddress=1433、packetDeltaCount=3、octetDeltaCount=152當作必要條件，隨機對某個上班日萃取出符合條件的 Netflow，然後以 sourceIPAddress 做彙整 Bucket，然後每個 Bucket 內再針對 destinationIPAddress 與 sourceTransportPort 做彙整，計算每個 sourceIPAddress 以這種特徵連線了多少個 destinationIPAddress，ES 的 Query 語法如下：

```
{
  "query":{
    "bool":{
      "filter":{
        "bool":{
          "must":[
            {"match":{"destination-
TransportPort":1433}},
            {"match":{"packetDeltaCount":3}},
            {"match":{"octetDeltaCount":152}}
          ]
        }
      }
    }
  },
  "aggs":{
    "AggsIp":{
      "terms":{
        "field":"sourceIPAddress",
        "size":100000,
        "order":{"CountDst":"desc"}
      },
      "aggs":{
        "CountDst":{
          "cardinality":{
            "field":"destinationIPAddress"
          }
        },
        "CountPort":{
          "cardinality":{
            "field":"sourceTransportPort"
          }
        }
      }
    },
    "CountIP":{
      "cardinality":{
        "field":"sourceIPAddress"
      }
    }
  }
}
```

上述查詢得到結果有286個 IP 以此特徵對外連線，排序依據是連線目標 IP 數量，從多到少，本文限於篇幅僅擷取前幾名如下**表 1**：

表 1 特徵1433-3-52搜索 Netflow 結果

來源 IP	目標 IP 數量	Netflow 筆數	來源 Port 數
1.170.36.229	63788	73274	13409
140.130.93.101	10076	31483	6936
122.165.92.180	6238	6542	5698
113.16.174.66	1295	1577	1318
202.127.28.58	1166	1556	1166
140.130.46.118	753	1277	753

從上表我們可以看得出來它們都有問題，連線 IP 數量近似於 Netflow 筆數，亦即它們對每個目標幾乎只建立一次 TCP Session，通常應用如 HTTP、MAIL 等等總是會對同一個 Server 建立多次 TCP 連線的，所以它們的行為類似只為了探測而且不重複連線，第四欄統計 Source-Port 數量是為了確認該 IP 是否伺服器回應 Client 的傳輸(這種情況 Source-Port 數應該只有1個，就是 Service-Port)，但很顯然它們也都不是伺服器的回應 Netflow。而上表所列前幾名，除了第一名超過告警臨界值會被注意，第二名之後量都很小，整天下來才幾千筆，資安設備根本不會感覺異常，然而它們確實是有問題的，當我們繼續分析前後幾日資料，發現這些異常 IP 的雷同度相當高，而且整日數量也都是一兩千，真正的細水長流，屬於 Botnet Master 口袋內的預備兵，而這些小規模的異常者正是本文研究的目標，經由上述觀察，我們已經確立尋找小規模異常者的一種方法，目前為止我們所採用的步驟都可以依靠程式來自動化運行，不斷地累積這些異常者參與過的特徵到資料庫中。

接下來的研究方向就是如何分類分群，如何利用這些 Botnet Master 口袋中的棋子來獲得更多 Botnet 的資訊，例如分享資料庫給學校網管以及有興趣的研究室，如果學校網管能對這些異常機器做長期封包追蹤分析，或許能截獲上游 C&C 資訊與 Malware。

當然，隨意亂堆的資料會變成垃圾，因此我們也需要研究如何保存累積這些資料才能有助於後續的研究，現階段的作法是以日為單位，以特徵為關鍵值，每日參與某特徵的異常者資料存為一個 ES 的 Document，未來可以針對特徵做查詢，也可以針對 IP 做查詢，此外，由於 ES 做某個欄位的總表陣列會耗費較多記憶體，為了精省計算資源，必須在傳統 SQL 資料庫同步維護一個特徵總表與異常 IP 總表，雖然程式上比較麻煩，但卻可以節省很多 ES 的運算資源，降低崩潰風險。

1.4 結論與未來工作

在本文中我們提到如何優化運算資源的使用效率，也實作定期自動化從大規模異常者萃取行為特徵，再去全骨幹 Netflow 資料中搜索相同特徵的小規模異常者，將異常者記入資料庫逐漸累積，目前我們僅實作出 Port-Packet-Octet 這一種特徵的自動化小規模 Botnet 查找，未來我們會繼續開發更多特徵查找方法，例如 Port 與多組 Packet-Octet 特徵做近似搜索的方法。

做為骨幹網路 NOC，我們最需要替使用者防護的惡意行為就是 DDoS，對於假冒來源 IP 的攻擊方式，如果在最靠近受害者端的骨幹介面封鎖，會連無辜被冒充者的正常連線也被封鎖，因此我們比較傾向於「緩解壓力」的作法，異常偵測系統會在告警時實時統計異常 Flow 經過的骨幹網路介面，回溯邊界入口並排序，只在規模較大的入口封鎖，可以減輕受害者壓力，也可能減少誤傷無辜者傳輸的機率。至於 Reflective 的攻擊方式，該如何讓程式可以自動從 Netflow 中判斷來源 IP 是被 Reflective 的無辜者，目前我們還在研究中。

我們也將研究針對正常行為 Netflow 做機器學習，建立正常使用的 Netflow 行為模型，未來如果我們優化效率更好，或是具備更多運算資源時，會將不符合正常使用行為模型的 IP 都拉出來做異常判斷，總而言之，如何判斷異常並非是唯一的路，反過來判斷是否正常或許更加的簡單。

參考文獻

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

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

[3] https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html

[4] https://www.speedguide.net/