通過Firepower服務處理單流大型會話（大象流）

目錄

簡介

本文說明為什麼單個流量不能消耗Cisco Firepower裝置的整個額定吞吐量。

背景資訊

任何頻寬速度測試網站的結果或任何頻寬測量工具（例如iperf）的輸出可能未顯示Cisco Firepower裝置的通告吞吐量等級。同樣，通過任何傳輸協定傳輸非常大的檔案並不能證明Firepower裝置的通告吞吐量級別。發生這種情況的原因是Firepower服務不使用單個網路流來確定其最大吞吐量。

按Snort處理流量

Firepower服務的底層檢測技術是Snort。在Cisco Firepower裝置上實施Snort是一個單執行緒進程，用於處理流量。根據通過裝置的所有流的總吞吐量，對裝置進行特定評分。預計這些裝置將部署在企業網路上，通常在邊界邊緣附近，並且可處理數千個連線。

Firepower服務使用在裝置上每個CPU上運行的一個Snort進程對多個不同的Snort進程進行流量負載均衡。理想情況下，系統會在所有Snort進程中均衡地平衡流量。Snort需要能夠為下一代防火牆(NGFW)、入侵防禦系統(IPS)和高級惡意軟體防護(AMP)檢測提供適當的情景分析。為了確保Snort最有效，會將來自單個流的所有流量負載均衡到一個Snort實例。如果來自單個流的所有流量未均衡到單個Snort實例，則可以避開系統，並且流量將會以某種方式溢位，使得Snort規則可能不太匹配或者對於AMP檢查而言某個檔案的片段並非連續。因此，負載平衡演算法基於能夠唯一地標識給定連線的連線資訊。

具備Firepower服務和NGIPS虛擬的ASA中的二元組演算法

在具備Firepower服務平台和下一代入侵防禦系統(NGIPS)虛擬的自適應安全裝置(ASA)上，流量會進行負載均衡，以便使用二元組演算法進行Snort。此演算法的資料點為：

- 來源 IP
- 目的地 IP

Firepower和FTD裝置上軟體版本5.3或更低版本中的3元組演算法

在所有先前版本（5.3或更低版本）上，流量會負載均衡到使用3元組演算法的Snort。此演算法的資料點為：

- 來源 IP
- 目的地 IP
- IP通訊協定

任何具有相同源、目標和IP協定的流量都會被負載均衡到同一個Snort例項。

Firepower和FTD裝置上軟體版本5.4、6.0及更高版本中的5元組演算法

在5.4、6.0或更高版本中，使用5元組演算法將流量負載均衡到Snort。所考慮的資料點包括：

- 來源 IP
- 來源連線埠
- 目的地 IP
- 目的地連線埠
- IP通訊協定

將埠新增到演算法的目的是當特定源和目標對佔據大部分流量時，更加平均地平衡流量。通過新增埠，高位臨時源埠的每個流量必須不同，並且必須更平均地新增額外的熵，以將流量平衡到不同的snort例項。

總吞吐量

裝置的總吞吐量是根據所有發揮最大潛能的Snort例項的總吞吐量來衡量的。測量吞吐量的行業標準做法適用於具有不同對象大小的多個HTTP連線。例如，NSS NGFW測試方法測量具有44k、21k、10k、4.4k和1.7k對象的裝置的總吞吐量。由於HTTP連線中涉及的其他資料包，這些轉換轉換為平均資料包大小範圍，從大約1k位元組到128位元組。

您可以估計單個Snort例項的效能等級。將裝置的額定吞吐量除以運行的Snort例項數。例如，對於平均資料包大小為1k位元組的IPS，如果裝置的額定值為10Gbps，並且該裝置有20個Snort例項，則單個例項的最大吞吐量近似為每個Snort的500 Mbps。不同型別的流量、網路協定、資料包大小以及整體安全策略的差異都可能會影響觀察到的裝置吞吐量。

第三方工具測試結果

當您使用任何速度測試網站或任何頻寬測量工具（例如iperf）進行測試時，將生成一個大型單流TCP流。這種大型TCP流稱為大象流。Elephant Flow是單個會話，運行時間相對較長，佔用大量或不成比例頻寬的網路連線。這種型別的流分配給一個Snort例項，因此測試結果顯示單個snort例項的吞吐量，而不是裝置的聚合吞吐量級別。

觀察到的症狀

觀察到的高CPU

Elephant Flows的另一個可見影響可能是snort例項高cpu。這可透過「show asp inspect-dp snort」或shell「top」工具看到。

```
> show asp inspect-dp snort

SNORT Inspect Instance Status Info

Id  Pid   Cpu-Usage  Conns   Segs/Pkts  Status tot (usr | sys)
--  ----  ---------  -----   --------   ----------

0  48500  30% ( 28%|  1%)  12.4 K      0        READY
1  48474  24% ( 22%|  1%)  12.4 K      0        READY
2  48475  34% ( 33%|  1%)  12.5 K      1        READY
3  48476  29% ( 28%|  0%)  12.4 K      0        READY
4  48477  32% ( 30%|  1%)  12.5 K      0        READY
5  48478  31% ( 29%|  1%)  12.3 K      0        READY
6  48479  29% ( 27%|  1%)  12.3 K      0        READY
7  48480  23% ( 23%|  0%)  12.2 K      0        READY
8  48501  27% ( 26%|  0%)  12.6 K      1        READY
9  48497  28% ( 27%|  0%)  12.6 K      0        READY
10 48482  28% ( 27%|  1%)  12.3 K      0        READY
11 48481  31% ( 30%|  1%)  12.5 K      0        READY
12 48483  36% ( 36%|  1%)  12.6 K      0        READY
13 48484  30% ( 29%|  1%)  12.4 K      0        READY
14 48485  33% ( 31%|  1%)  12.6 K      0        READY
15 48486  38% ( 37%|  0%)  12.4 K      0        READY
16 48487  31% ( 30%|  1%)  12.4 K      1        READY
17 48488  37% ( 35%|  1%)  12.7 K      0        READY
18 48489  34% ( 33%|  1%)  12.6 K      0        READY
19 48490  27% ( 26%|  1%)  12.7 K      0        READY
20 48491  24% ( 23%|  0%)  12.6 K      0        READY
21 48492  24% ( 23%|  0%)  12.6 K      0        READY
22 48493  28% ( 27%|  1%)  12.4 K      1        READY
23 48494  27% ( 27%|  0%)  12.2 K      0        READY
24 48495  29% ( 28%|  0%)  12.5 K      0        READY
25 48496  30% ( 30%|  0%)  12.4 K      0        READY
26 48498  29% ( 27%|  1%)  12.6 K      0        READY
27 48517  24% ( 23%|  1%)  12.6 K      0        READY
28 48499  22% ( 21%|  0%)  12.3 K      1        READY
29 48518  31% ( 29%|  1%)  12.4 K      2        READY
30 48502  33% ( 32%|  0%)  12.5 K      0        READY

31 48514 80% ( 80%| 0%) 12.7 K 0 READY <<< CPU 31 is much busier than the rest, and will stay
busy for while with elephant flow.

32 48503  49% ( 48%|  0%)  12.4 K      0        READY
33 48507  27% ( 25%|  1%)  12.5 K      0        READY
34 48513  27% ( 25%|  1%)  12.5 K      0        READY
35 48508  32% ( 31%|  1%)  12.4 K      0        READY
36 48512  31% ( 29%|  1%)  12.4 K      0        READY



$ top
```

```
PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND

69470 root       1 -19 9088m 1.0g  96m R   80  0.4 135:33.51 snort     <<<< one snort very busy,
rest below 50%

69468 root       1 -19 9089m 1.0g  99m R   49  0.4 116:08.69 snort
69467 root       1 -19 9078m 1.0g  97m S   47  0.4 118:30.02 snort
69492 root       1 -19 9118m 1.1g  97m R   47  0.4 116:40.15 snort
69469 root       1 -19 9083m 1.0g  96m S   39  0.4 117:13.27 snort
69459 root       1 -19 9228m 1.2g  97m R   37  0.5 107:13.00 snort
69473 root       1 -19 9087m 1.0g  96m R   37  0.4 108:48.32 snort
69475 root       1 -19 9076m 1.0g  96m R   37  0.4 109:01.31 snort
69488 root       1 -19 9089m 1.0g  97m R   37  0.4 105:41.73 snort
69474 root       1 -19 9123m 1.1g  96m S   35  0.4 107:29.65 snort
69462 root       1 -19 9065m 1.0g  99m R   34  0.4 103:09.42 snort
69484 root       1 -19 9050m 1.0g  96m S   34  0.4 104:15.79 snort
69457 root       1 -19 9067m 1.0g  96m S   32  0.4 104:12.92 snort
69460 root       1 -19 9085m 1.0g  97m R   32  0.4 104:16.34 snort
```

使用上述的5-Tuple演算法，將始終向同一個snort例項傳送長壽命流。如果snort中有大量的AVC、IPS、檔案等策略處於活動狀態，則在一段時間內在snort例項上可以看到CPU使用率較高(>80%)。新增SSL策略會進一步增加CPU使用率，從而導致SSL解密的計算成本很高。

許多Snort CPU中少數的CPU使用率較高不會導致嚴重警報。它是NGFW系統在對流執行深度資料包檢查時的行為，並且這可以自然地使用大部分CPU。一般來說，NGFW不會處於嚴重的CPU耗盡狀態，直到大多數snort CPU超過95%並保持超過95%且資料包丟棄情況出現。

以下補救措施有助於解決因大象流而導致的高CPU問題。

補救

智慧型應用程式略過 (IAB)

軟體版本6.0引入了一項稱為IAB的新功能。當Firepower裝置達到預定義效能閾值時，IAB功能會查詢符合特定標準的流，以便智慧地繞過以減輕檢測引擎壓力的流。

> **提示：**有關IAB配置的詳細資訊，請參閱。

識別並信任大資料流

大流量通常與高使用率低檢測值流量有關，例如備份、資料庫複製等。這些申請中有許多不能從檢查中受益。為了避免大流量問題，您可以識別大流量並為它們建立訪問控制信任規則。這些規則能夠唯一地識別大型流，允許這些流未經檢查就通過，並且不受單個snort例項行為的限制。

> **附註**：若要識別信任規則的大流量，請聯絡Cisco Firepower TAC。

相關資訊

- 使用智慧應用旁路進行訪問控制

- 技術支援與文件 - Cisco Systems