Process Single Stream Large Session (Elephant Flow) by Firepower Services

Contents

Introduction

This document describes why a single flow cannot consume the entire rated throughput of a Cisco Firepower appliance.

Background Information

The result of any bandwidth speed testing website, or the output of any bandwidth measurement tool (for example, iperf) might not exhibit the advertised throughput rating of the Cisco Firepower appliances. Similarly, the transfer of a very large file over any transport protocol does not demonstrate the advertised throughput rating of a Firepower appliance. It occurs because the Firepower service does not use a single network flow in order to determine its maximum throughput.

Process Traffic by Snort

The underlying detection technology of the Firepower service is Snort. The implementation of Snort on the Cisco Firepower appliance is a single thread process in order to process traffic. An appliance is rated for a specific rating based on the total throughput of all flows that goes through the appliance. It is expected that the appliances are deployed on a Corporate network, usually near the border edge and works with thousands of connections.

Firepower Services use load balancing of traffic to a number of different Snort process with one Snort process that runs on each CPU on the appliance. Ideally, the system load balances traffic evenly across all of the Snort processes. Snort needs to be able to provide proper contextual analysis for Next-Generation Firewall (NGFW), Intrusion Prevention System (IPS) and Advanced Malware Protection (AMP) inspection. In order to ensure Snort is most effective, all the traffic from a single flow is load balanced to one snort instance. If all the traffic from a single flow was not balanced to a single snort instance, the system could be evaded and the traffic would spilt in such a way that a Snort rule might be less likely to match or pieces of a file are not contiguous for AMP inspection. Therefore, the load balancing algorithm is based on the connection information that can uniquely identify a given connection.

2-Tuple Algorithm in ASA with Firepower Services and NGIPS Virtual

On the Adaptive Security Appliance (ASA) with Firepower Service platform and Next Generation Intrusion Prevention System (NGIPS) virtual, traffic is load balanced in order to Snort with the use of a 2-tuple algorithm. The datapoints for this algorithm are:

- Source IP
- Destination IP

3-Tuple Algorithm in Software Version 5.3 or Lower on Firepower and FTD Appliances

On all prior Versions (5.3 or lower), traffic is load balanced to Snort that uses a 3-tuple algorithm. The datapoints for this algorithm are:

- Source IP
- Destination IP
- IP Protocol

Any traffic with the same source, destination, and IP Protocol are load balanced to the same instance of Snort.

5-Tuple Algorithm in Software Version 5.4, 6.0, and Greater on Firepower and FTD Appliances

On Version 5.4, 6.0 or greater, traffic is load balaned to Snort with a 5-tuple algorithm. The datapoints that are taken into account are:

- Source IP
- Source Port
- Destination IP
- Destination Port
- IP Protocol

The purpose to add ports to the algorithm is to balance traffic more evenly when there are specific source and destination pairs that account for large portions of the traffic. By addition of the ports, the high order ephemeral source ports must be different per flow, and must add additional entropy more evenly that balances the traffic to different snort instances.

Total Throughput

The total throughput of an appliance is measured based on the aggregate throughput of all the snort instances that works to their fullest potential. Industry standard practices in order to measure the throughput are for multiple HTTP connections with various object sizes. For example, the NSS NGFW test methodology measures total throughput of the device with 44k, 21k, 10k, 4.4k, and 1.7k objects. These translate to a range of average packet sizes from around 1k & bytes to 128 bytes because of the other packets involved in the HTTP connection.

You can estimate the performance rating of an individual Snort instance. Take the rated throughput of the appliance and divide that by the number of Snort instances that run. For example, if an appliance is rated at 10Gbps for IPS with an average packet size of 1k bytes, and that appliance has 20 instances of Snort, the approximate maximum throughput for a single instance would be 500 Mbps per Snort. Different types of traffic, network protocols, sizes of the packets along with differences in the overall security policy can all impact the observed throughput of the device.

Third Party Tool Test Result

When you test with any speed testing website, or any bandwidth measurement tool, such as, iperf, one large single stream TCP flow is generated. This type of large TCP flow is called an Elephant Flow. An Elephant Flow is a single session, relatively long running network connection that consumes a large or disproportionate amount of bandwidth. This type of flow is assigned to one Snort instance, therefore the test result displays the throughput of single snort instance, not the aggregate throughput rating of the appliance.

Observed Symptoms

Observed High CPU

Another visible effect of Elephant Flows can be snort instance high cpu. This can be seen via "show asp inspect-dp snort", or with the shell "top" tool.

```
> 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
```

With 5-Tuple algorithm described above, a long lived flow will always be sent to the same snort instance. If there are extensive AVC, IPS, File, etc policies active in snort, the CPU can be seen high (>80%) on a snort instance for some period of time. Adding SSL policy will further increase CPU usage do to the computationally expensive nature of SSL Decryption.

High CPU on few of the many snort CPUs is not a cause for critical alarm. It is the behavior of the NGFW system in performing deep packet inspection into a flow, and this can naturally use large portions of a CPU. As a general guideline, the NGFW is not in a critical CPU starvation situation until most of the snort CPUs are over 95% and remain over 95% and packet drops are being seen.

The Remediations below will help with high CPU situation due to Elephant flows.

Remediations

Intelligent Application Bypass (IAB)

The software version 6.0 introduces a new feature called IAB. When a Firepower appliance reaches a pre-defined performance threshold, the IAB feature looks for flows that meet specific criteria in order to intelligently bypass that alleviates pressure on the detection engines.

> **Tip**: More information on the configuration of the IAB can be found here.

Identify and Trust Large Flows

Large flows are often related to high use low inspection value traffic for example, backups, database replication, etc. Many of these applications can not be benefited from inspection. In order to avoid issues with large flows, you can identify the large flows and create Access Control trust rules for them. These rules are able to uniquely identify large flows, allow those flows to pass uninspected, and not to be limited by the single snort instance behavior.

> **Note**: In order to identify large flows for trust rules, contact the Cisco Firepower TAC.

Related Information

- Access Control Using Intelligent Application Bypass
- Technical Support & Documentation - Cisco Systems