Firepower サービスによる単一ストリーム大規模セッション（エレファント フロー）の処理

内容

[概要](#)
[背景説明](#)
[Snortによるトラフィックの処理](#)
[ASA with Firepower ServicesおよびNGIPS仮想の2タプルアルゴリズム](#)
[FirepowerおよびFTDアプライアンスのソフトウェアバージョン5.3以下の3タプルアルゴリズム](#)
[FirepowerおよびFTDアプライアンスのソフトウェアバージョン5.4、6.0、およびそれ以降の5タプルアルゴリズム](#)
[スループット合計](#)
[サードパーティツールのテスト結果](#)
[観察された症状](#)
[高CPU使用率の確認](#)
[修正](#)
[インテリジェント アプリケーション バイパス（IAB）](#)
[大規模フローの特定と信頼](#)
[関連情報](#)

概要

このドキュメントでは、1つのフローでCisco Firepowerアプライアンスの定格スループット全体を消費できない理由について説明します。

背景説明

帯域幅速度テストのWebサイトの結果、または帯域幅測定ツール（iperfなど）の出力では、Cisco Firepowerアプライアンスのアドバタイズされたスループット評価が示されない場合があります。同様に、任意のトランスポートプロトコルで非常に大きなファイルを転送しても、Firepowerアプライアンスのアドバタイズされたスループット評価は示されません。これは、Firepowerサービスが最大スループットを決定するために単一のネットワークフローを使用していないために発生します。

Snortによるトラフィックの処理

Firepower サービスの基盤となる検出テクノロジーは Snort です。Cisco FirepowerアプライアンスでのSnortの実装は、トラフィックを処理するための1つのスレッドプロセスです。アプライアンスは、アプライアンスを通過するすべてのフローの総スループットに基づいて、特定の評価で評価されます。アプライアンスが社内ネットワーク（通常は境界エッジの近く）に展開されていること、および何千もの接続を処理することが想定されています。

Firepower Servicesは、アプライアンス上の各CPUで1つのSnortプロセスが実行される複数の異なるSnortプロセスへのトラフィックのロードバランシングを使用します。理想的には、システム

はすべてのSnortプロセス間でトラフィックを均等にロードバランシングします。Snortは、Next-Generation Firewall(NGFW)、侵入防御システム(IPS)、および高度なマルウェア防御(AMP)検査に対して適切なコンテキスト分析を提供できる必要があります。Snortが最も効果的であることを保証するため、1つのフローからのすべてのトラフィックが1つのSnortインスタンスにロードバランシングされます。1つのフローからのすべてのトラフィックが単一のSnortインスタンスにバランシングされなかった場合は、システムが回避され、AMP検査が不可。したがって、ロードバランシングアルゴリズムは、特定の接続を一意に識別できる接続情報に基づいています。

ASA with Firepower ServicesおよびNGIPS仮想の2タプルアルゴリズム

Firepower ServiceプラットフォームとNext Generation Intrusion Prevention System(NGIPS)仮想を備えた適応型セキュリティアプライアンス(ASA)では、2タプルアルゴリズムを使用してSnortへのトラフィックのロードバランシングが行われます。このアルゴリズムのデータポイントは次のとおりです。

- 送信元 IP
- 宛先 IP

FirepowerおよびFTDアプライアンスのソフトウェアバージョン5.3以下の3タプルアルゴリズム

以前のすべてのバージョン（5.3以下）では、トラフィックは3タプルアルゴリズムを使用するSnortにロードバランシングされます。このアルゴリズムのデータポイントは次のとおりです。

- 送信元 IP
- 宛先 IP
- IP プロトコル

送信元、宛先、および IP プロトコルがすべて同じであるトラフィックはすべて、同じ Snort インスタンスにロード バランシングされます。

FirepowerおよびFTDアプライアンスのソフトウェアバージョン5.4、6.0、およびそれ以降の5タプルアルゴリズム

バージョン5.4、6.0以降では、トラフィックは5タプルアルゴリズムを使用してSnortにロードバランスされます。考慮されるデータポイントは次のとおりです。

- 送信元 IP
- 送信元ポート
- 宛先 IP
- 宛先ポート
- IP プロトコル

アルゴリズムにポートを追加する目的は、トラフィックの大部分を占める特定の送信元と宛先のペアがある場合に、トラフィックのバランスをより均等にすることです。ポートを追加することで、上位の一時的な送信元ポートはフローごとに異なる必要があり、異なるSnortインスタンスにトラフィックを分散させるエントロピーを均等に追加する必要があります。

スループット合計

アプライアンスの総スループットは、最大限に機能するすべてのSnortインスタンスの総スループ

ットに基づいて測定されます。スループットを測定するための業界標準のプラクティスは、さまざまなオブジェクトサイズの複数のHTTP接続に対するものです。たとえば、NSS NGFWテスト手法は、44k、21k、10k、4.4k、および1.7kオブジェクトを使用してデバイスの総スループットを測定します。これらは、HTTP接続に関係するその他のパケットにより、平均パケットサイズが約1kバイトから128バイトまでの範囲に変換されます。

個々のSnortインスタンスのパフォーマンス評価を見積もることができます。アプライアンスの定格スループットを、実行するSnortインスタンスの数で割ります。たとえば、アプライアンスが平均パケットサイズが1kバイトのIPSで10 Gbpsと評価され、そのアプライアンスにSnortのインスタンスが20ある場合、1つのインスタンスの最大スループットはSnortあたり500 Mbpsになります。トラフィックの種類、ネットワークプロトコル、パケットのサイズ、セキュリティポリシー全体の違いなど、デバイスの観察されるスループットに影響を与える可能性があります。

サードパーティツールのテスト結果

速度をテストする Web サイトまたは帯域幅測定ツール（iperf など）でテストした場合、単一の大規模ストリーム TCP フローが生成されます。このタイプの大規模 TCP フローは、エレファント フローと呼ばれます。1 つのエレファント フローは単一のセッションであり、大量の（または不均衡な）帯域幅を消費する、比較的長期間実行されるネットワーク接続です。このタイプのフローは 1 つの Snort インスタンスに割り当てられるため、アプライアンスの総スループット速度ではなく、単一の Snort インスタンスのスループットがテスト結果に示されます。

観察された症状

高CPU使用率の確認

ゾウのフローのもう1つの目に見える影響は、snortインスタンスの高いcpuです。これは、「show asp inspect-dp snort」またはシェルの「top」ツールで確認できます。

```
> 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
```

上で説明した5タプルのアルゴリズムでは、長く続いたフローが常に同じSnortインスタンスに送信されます。Snortでアクティブな広範なAVC、IPS、ファイルなどのポリシーが存在する場合、SnortインスタンスのCPU使用率が一定期間高くなる（80 %を超える）ことがあります。SSLポリシーを追加すると、SSL復号化の計算コストの高い性質に対するCPU使用率がさらに増加します。

多数のSnort CPUの中の一部でCPUの使用率が高くなることは、クリティカルアラームの原因ではありません。これは、フローにディープパケットインスペクションを実行する際のNGFWシステムの動作であり、これは当然、CPUの大部分を使用できます。一般的なガイドラインとして、NGFWは、ほとんどのSnort CPUが95 %を超え、95 %を超え、パケットのドロップが見られるまで、重大なCPUスタベーション状態にはなりません。

以下の修復は、ゾウのフローによるCPUの高使用状態に役立ちます。

修正

インテリジェント アプリケーション バイパス (IAB)

ソフトウェアバージョン6.0では、IABという新しい機能が導入されています。Firepowerアプライアンスが事前定義されたパフォーマンスしきい値に達すると、IAB機能は特定の基準を満たすフローを探し出し、検出エンジンの圧力を軽減するインテリジェントなバイパスを行います。

> **ヒント**：IABの設定の詳細については、こちらを参照してください。

大規模フローの特定と信頼

大量のフローは、バックアップ、データベースのレプリケーションなど、使用率の低い検査値のトラフィックに関連することがよくあります。これらのアプリケーションの多くは、検査の恩恵を受けることはできません。大きなフローに関する問題を回避するには、大きなフローを特定し、それらのフローに対するアクセスコントロール信頼ルールを作成します。これらのルールは、大きなフローを一意に識別し、それらのフローが検査されずに通過できるようにし、単一のSnortインスタンスの動作によって制限されないようにすることができます。

> **注**：信頼ルールの大きなフローを特定するには、Cisco Firepower TACに連絡してください。

関連情報

- インテリジェント アプリケーション バイパスを使用したアクセス制御
- テクニカル サポートとドキュメント – Cisco Systems