FirePOWER Services의 단일 스트림 대형 세션(코끼리 흐름) 처리

목차

소개

이 문서에서는 단일 플로우가 Cisco Firepower 어플라이언스의 전체 정격 처리량을 소비하지 못하는 이유를 설명합니다.

배경 정보

대역폭 속도 테스트 웹 사이트 또는 대역폭 측정 툴(예: iperf)의 결과 Cisco Firepower 어플라이언스의 알려진 처리량 등급이 표시되지 않을 수 있습니다.마찬가지로, 어떤 전송 프로토콜로도 매우 큰 파일을 전송하더라도 Firepower 어플라이언스의 알려진 처리량 등급을 시연하지 않습니다.이는 Firepower 서비스가 최대 처리량을 결정하기 위해 단일 네트워크 흐름을 사용하지 않기 때문에 발생합니다.

Snort별 트래픽 처리

Firepower 서비스의 기본 탐지 기술은 Snort입니다.Cisco Firepower 어플라이언스에서 Snort를 구현하는 것은 트래픽을 처리하기 위한 단일 스레드 프로세스입니다.어플라이언스는 어플라이언스를 통과하는 모든 플로우의 총 처리량을 기준으로 특정 등급을 평가합니다.어플라이언스는 대개 경계 엣지 근처에 있는 기업 네트워크에 구축되어 수천 개의 연결과 연동될 것으로 예상됩니다.

Firepower Services는 어플라이언스의 각 CPU에서 실행되는 하나의 Snort 프로세스를 통해 여러 다른 Snort 프로세스에 대한 트래픽 로드 밸런싱을 사용합니다.이상적으로, 시스템은 모든 Snort 프로세스 전반에 걸쳐 트래픽을 균등하게 로드 밸런싱합니다.Snort는 NGFW(Next-Generation Firewall), IPS(Intrusion Prevention System) 및 AMP(Advanced Malware Protection) 검사를 위한 적절한 상황 분석을 제공할 수 있어야 합니다.Snort가 가장 효과적인지 확인하기 위해 단일 플로우

의 모든 트래픽이 단일 Snort 인스턴스로 로드 밸런싱됩니다. 단일 플로우의 모든 트래픽이 단일 Snort 인스턴스로 밸런싱되지 않은 경우 시스템을 우회할 수 있으며 Snort 규칙이 일치할 가능성이 적거나 파일의 일부가 AMP 검사를 위해 연속되지 않도록 트래픽을 분산할 수 있습니다.따라서 로드 밸런싱 알고리즘은 지정된 연결을 고유하게 식별할 수 있는 연결 정보를 기반으로 합니다.

ASA with Firepower Services 및 NGIPS Virtual의 2-튜플 알고리즘

Firepower Service 플랫폼과 NGIPS(Next Generation Intrusion Prevention System) 가상 환경의 ASA(Adaptive Security Appliance)에서 2튜플 알고리즘을 사용하여 Snort에 트래픽을 로드 밸런싱합니다.이 알고리즘의 데이터 포인트는 다음과 같습니다.

- 소스 IP
- 대상 IP

Firepower 및 FTD 어플라이언스에서 소프트웨어 버전 5.3 이하의 3-튜플 알고리즘

모든 이전 버전(5.3 이하)에서 트래픽은 3-튜플 알고리즘을 사용하는 Snort로 로드 밸런싱됩니다.이 알고리즘의 데이터 포인트는 다음과 같습니다.

- 소스 IP
- 대상 IP
- IP 프로토콜

소스, 대상 및 IP 프로토콜이 동일한 모든 트래픽은 동일한 Snort 인스턴스로 로드 밸런싱됩니다.

Firepower 및 FTD 어플라이언스에서 소프트웨어 버전 5.4, 6.0 이상의 5-튜플 알고리즘

버전 5.4, 6.0 이상에서 트래픽은 5-튜플 알고리즘을 사용하여 Snort에 로드 밸런싱됩니다.고려해야 하는 데이터 포인트는 다음과 같습니다.

- 소스 IP
- 소스 포트
- 대상 IP
- 대상 포트
- IP 프로토콜

알고리즘에 포트를 추가하는 목적은 트래픽의 많은 부분을 차지하는 특정 소스 및 목적지 쌍이 있을 때 트래픽의 균형을 더 균등하게 조정하는 것입니다.포트 외에도, 높은 순서의 임시 소스 포트는 흐름마다 달라야 하며, 트래픽을 서로 다른 Snort 인스턴스로 밸런싱하기 위해 엔트로피를 더 고르게 추가해야 합니다.

총 처리량

어플라이언스의 총 처리량은 최대 잠재력을 발휘하는 모든 Snort 인스턴스의 총 처리량을 기준으로 측정됩니다.처리량을 측정하기 위한 업계 표준 관행은 다양한 객체 크기의 다중 HTTP 연결에 대한 것입니다.예를 들어, NSS NGFW 테스트 방법론은 44k, 21k, 10k, 4.4k, 1.7k object로 디바이스의 총 처리량을 측정합니다.이러한 패킷은 HTTP 연결과 관련된 다른 패킷 때문에 약 1k 및 바이트 ~128바이트의 평균 패킷 크기 범위로 변환됩니다.

개별 Snort 인스턴스의 성능 등급을 추정할 수 있습니다.어플라이언스의 정격 처리량을 실행하고

실행하는 Snort 인스턴스의 수로 나눕니다.예를 들어, 어플라이언스가 평균 패킷 크기가 1k 바이트인 IPS에 대해 10Gbps로 평가되고 해당 어플라이언스에 Snort 인스턴스가 20개 있는 경우 단일 인스턴스의 대략적인 최대 처리량은 Snort당 500Mbps입니다.트래픽 유형, 네트워크 프로토콜, 패킷의 크기 및 전반적인 보안 정책의 차이는 모두 디바이스의 관찰된 처리량에 영향을 미칠 수 있습니다.

서드파티 툴 테스트 결과

속도 테스트 웹 사이트 또는 iperf와 같은 대역폭 측정 도구를 사용하여 테스트할 경우 하나의 큰 단일 스트림 TCP 흐름이 생성됩니다.이러한 유형의 대규모 TCP 흐름을 Elephant Flow라고 합니다.Elephant Flow는 비교적 오랫동안 실행되는 단일 세션으로서, 대역폭이 크거나 지나치게 많이 사용됩니다.이 유형의 플로우는 하나의 Snort 인스턴스에 할당되므로 테스트 결과에 어플라이언스의 총 처리량 등급이 아닌 단일 snort 인스턴스의 처리량이 표시됩니다.

관찰된 증상

높은 CPU 관찰

Elephant Flows의 또 다른 가시적인 효과는 Snort 인스턴스 high cpu일 수 있습니다.이는 "show asp inspect-dp snort" 또는 셸 "top" 도구를 통해 확인할 수 있습니다.

```
> 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
```

위에서 설명한 5-튜플 알고리즘을 사용하면 오래 지속되는 흐름이 항상 동일한 snort 인스턴스로 전송됩니다.snort에서 활성화된 광범위한 AVC, IPS, 파일 등의 정책이 있는 경우, CPU는 일정 기간 동안 Snort 인스턴스에서 높은(>80%)로 표시될 수 있습니다.SSL 정책을 추가하면 SSL 해독의 계산 비용이 많이 드는 특성에 대한 CPU 사용량이 더욱 증가합니다.

많은 Snort CPU 중 일부에서 높은 CPU를 사용하는 것은 심각한 경보의 원인이 아닙니다.NGFW 시스템은 플로우에 대한 심층 패킷 검사를 수행하며, 이는 자연스럽게 CPU의 많은 부분을 사용할 수 있습니다.일반적으로 NGFW는 대부분의 Snort CPU가 95% 이상, 95% 이상 유지되며 패킷 드롭이 표시될 때까지 심각한 CPU 기아 상황이 아닙니다.

아래의 교정은 코끼리 흐름으로 인해 CPU 사용량이 많은 경우에 도움이 됩니다.

교정

IAB(Intelligent Application Bypass)

소프트웨어 버전 6.0에는 IAB라는 새로운 기능이 도입되었습니다.Firepower 어플라이언스가 미리 정의된 성능 임계값에 도달하면 IAB 기능은 특정 기준을 충족하는 플로우를 검색하여 탐지 엔진의 부담을 줄여 지능적으로 우회합니다.

팁:IAB 컨피그레이션에 대한 자세한 내용은 여기를 참조하십시오.

대규모 플로우 식별 및 신뢰

대규모 플로우는 대개 백업, 데이터베이스 복제 등과 같이 사용량이 적은 검사 값 트래픽과 관련이 있습니다.이러한 애플리케이션 중 다수는 검사로부터 혜택을 받을 수 없습니다.큰 플로우의 문제를 피하려면 큰 플로우를 식별하고 해당 플로우에 대한 액세스 제어 신뢰 규칙을 생성할 수 있습니다 .이러한 규칙은 대규모 흐름을 고유하게 식별하고, 이러한 플로우가 검사되지 않은 상태로 통과하도록 허용하며, 단일 snort 인스턴스 동작으로 제한되지 않습니다.

참고:신뢰 규칙에 대한 큰 흐름을 식별하려면 Cisco Firepower TAC에 문의하십시오.

관련 정보

- 지능형 애플리케이션 우회를 사용한 액세스 제어
- 기술 지원 및 문서 – Cisco Systems