Snort와 Suricata의 탐지 기능과 성능에 대한 비교 연구★

정명기* · 안성진** · 박원형***

요 약

본 논문에서는 널리 사용되는 침입 탐지 시스템인 Snort와 Suricata에 대해서 탐지 기능 측면과 성능 측면을 비교해 보고자 하였다. 구체적으로 Snort와 비교해보았을 때 Suricata에는 추가된 탐지 기능과 새로 도입된 멀티 스레딩이 패킷 처리 속도에 가져다 준 변화에 대해 분석해보고자 하였다. 그 결과, Suricata에는 기존의 Snort에서는 존재하지 않았던 Protocol Identification과 HTTP Normalizer & Parser, 그리고 File Identification 기능이 추가되었다는 점을 발견할 수 있었다. 또한, 양적 처리 성능 측면에서도 Suricata의 경우 작동하는 CPU Core의 개수가 늘어날수록 Snort와의 처리 성능(PPS, Packets Per Second)의 차이가 벌어지는 것으로 나타났다. 따라서 이러한 점을 볼 때, Suricata는 양적/질적 측면에서 모두 Snort보다 개선된 것으로 나타났기 때문에 Snort의 대안으로 사용되기에 적절하다는 결론을 내릴 수 있었다.

A Comparative Study on Function and Performance of Snort and Suricata

Myeong Ki Jeong* · Seongjin Ahn** · Won Hyung Park***

ABSTRACT

We have tried to compare two different IDSs which are widespread over the network administrator, Snort and Suricata, in functional and performance aspects. Specifically, we focused on analyzing upon what functions for detecting threat were added newly and what Multi-Threading introduced newly for Suricata has influenced in a performance aspect. As a result, we could discover that there are some features in Suricata which has never existed in Snort such as Protocol Identification, HTTP Normalizer & Parser, and File Identification. Also, It was proved that the gap of PPS(Packets Per Second) becomes wider, as the number of CPU Cores which are working increase. Therefore, we could conclude that Suricata can be an efficient alternative for Snort considering the result that Suricata is more effective quantitatively as well as qualitatively.

Key words : Open Source NIDS, Snort, Suricata

접수일(2014년 6월 27일), 수정일(1차: 2014년 9월 17일), 게재확정일(2014년 9월 28일)

★ 본 논문은 2014년 ETRI부설국가보안기술연구소 연구지원으로 수행되었음.

* 성균관대학교 컴퓨터교육과

** 성균관대학교 컴퓨터교육과

*** 극동대학교 사이버안보학과

1. 서 론

네트워크 보안에 대한 위협이 점점 증가하고 있는 추세가 나타남에 따라 이에 따른 보안 위협에 대한 대비 조치 역시 필요해지고 있는 실정이다. 이에 따라 네트워크 환경에 대한 위협을 미리 탐지해내고 능동적으로 대처하는 작업을 원활하게 수행하고자 하는 차원에서 여러 가지 보안 솔루션이 고안되고 개발되었는데, 그 중 하나가 침입탐지시스템(IDS, Intrusion Detection System)인 Snort이다.

Snort는 오픈 소스 환경에서 누구나 개발에 참여할 수 있도록 되어 있는 침입탐지시스템의 일종으로 시스템 내부에 내재된 규칙을 기반으로 규칙에 명시된 사항과 일치하는 패킷을 찾아내는 방식으로 침입탐지를 수행하는 IDS이다. 이러한 Snort는 1998년 세상에 모습을 드러낸 이후로 지금까지 오랜 기간 널리 사용되고 있다. [1] (Jay Beale, James C. Foster, Jeffrey Posluns, Brian Caswell, '스노트 2.0 마술상자', 에이콘, p.30, 2003.)

그런데, 최근 Snort의 대안으로서 Suricata라는 새로운 침입탐지시스템이 등장하게 되었다. Suricata는 기존의 Snort와 호환이 되면서 동시에 Snort보다 양적으로 더 좋은 처리 능률을 보이면서 질적으로도 더 다양한 방식의 패킷들을 탐지해낼 수 있는 침입탐지시스템을 만들어내고자 하는 목적으로 개발된 것으로, 기존의 Snort보다 효과 측면에서 더 월등한 침입탐지 작업을 수행하는 것을 목표로 하고 있다.

따라서 본 연구에서는 이러한 점에 착안하여 Snort와 Suricata의 특징에 대하여 정리하고 Snort와 Suricata의 양적 처리 속도와 질적 탐지 기능에 대하여 비교 분석하고, Suricata가 과연 Snort의 대안으로서 적절한지에 대해 알아보도록 하겠다.

2. Snort와 Suricata

Snort와 Suricata는 모두 오픈 소스(Open Source) 기반으로 동작하는 오픈 소스 NIDS(Open Source Network-Based Intrusion Detection System)이다. 다시 말하자면, 오픈 소스 환경에서 개발된 침입탐지시스템으로 누구나 자유롭게 사용할 수 있도록 소스가 공개되어 있으므로 사용자가 개발자가 될 수도 있도록 되어 있다는 것이다.

이렇듯 Snort와 Suricata는 Open Source 기반이라는 공통적인 특징을 갖고 있지만 동시에 세부적인 특징에서는 일정 부분 차이를 보이고 있기도 하다.

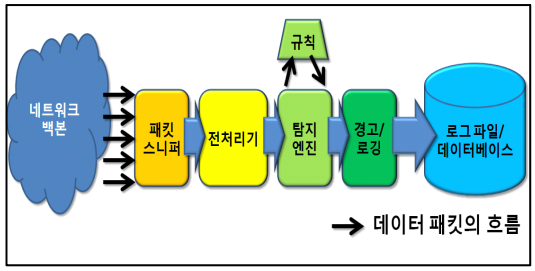
따라서 본 연구에서는 앞서 서론에서 언급했던 것과 같이 Snort와 Suricata의 비교 분석 작업을 수행하고자 한다. 그런데 이를 위해서는 Snort와 Suricata의 특징과 동작 방식에 대한 파악이 우선되어야 할 필요성이 있다. 따라서 본 장에서는 본격적인 비교 분석 작업을 수행하기에 앞서 Snort와 Suricata에 대해서 주요 특징을 각각 정리해보도록 하겠다.

2.1 Snort의 특징

Snort는 IDS 중에서 가장 널리 사용되는 시스템의 하나로, 오픈 소스 네트워크 기반 침입 탐지 시스템(Open Source NIDS)이다. Snort는 네트워크 기반 침입 탐지 시스템의 하나로 시스템 내부에 규칙을 갖고 이러한 내부 규칙과 시스템으로 들어오는 트래픽 패킷을 비교해보는 과정을 통해서 침입을 탐지해내도록 하고 있다.

Snort는 패킷 스니핑과 패킷 로깅 그리고 침입 탐지를 할 수 있는 기능을 갖추고 있다. 이러한 다양한 기능을 바탕으로 Snort는 침입 탐지의 역할을 수행한다고 할 수 있는데, Snort의 침입 탐지 역할에 대해서 좀 더 잘 이해하기 위해서는 Snort가 갖추고 있는 구조에 대해서 눈여겨보아야 한다.

Snort의 내부 동작 단계에 따른 구성 요소를 들여다보면, Snort는 스니퍼(Sniffer), 전처리기(Preprocessor), 탐지 엔진, 경고/로깅으로 정리해볼 수 있다.



(그림 1) Snort의 내부 동작 단계

우선 패킷 스니퍼에서 네트워크 백본으로부터 패킷을 받아들이는 것부터 시작된다. 이어서, 전처리기(Preprocessor)에서는 패킷 스니퍼에서 받아들인 것이 올바른 패킷인지에 대해서 판별하는 과정을 거치도록 한다. 다음으로는 탐지 엔진을 통해서 IDS 내부에 저장되어 있는 규칙과 패킷을 비교하는 과정을 통해서 침입을 탐지하도록 한다. 그리고 마지막 단계로 탐지 엔진에서의 결과를 바탕으로 네트워크 관리자에게 침입에 대한 경보를 발생시키거나 로그를 기록하도록 하여 로그 파일과 데이터베이스의 형태로 탐지 기록을 저장하도록 한다. [1] (Jay Beale, James C. Foster, Jeffrey Posluns, Brian Caswell, '스노트 2.0 마술상자', 에이콘, pp.34~40, 2003.)

2.2 Suricata의 특징

2009년 미국 국토안보부(US Department of Homel and Security)가 자금을 제공하여 비영리단체인 OISF (Open Information Security Foundation)를 새롭게 설립하였다.

미국 국토안보부의 자금 제공을 받은 OISF는 2010년 기존 Snort의 대안으로서 Suricata를 세상에 내놓았다.

이렇듯이 Snort의 대안으로 개발된 Suricata는 Snort를 근간으로 개발된 것으로 Snort의 내부 동작 방식과 유사하고 Snort에서 사용되었던 Snort VRT Ruleset, ET(Emerging Threat)-Open Ruleset, 그리고 ET-Pro Ruleset이 Suricata에서도 호환된다. [2]

<표 1> Snort와 Suricata의 Ruleset

IDS / Ruleset	Snort	Suricata
Snort VRT Rules	○	○
SO Rules	○	X
Emerging Threat Rules	○	○

다만, Snort의 대안으로 개발된 Suricata는 Snort와는 달리 멀티 스레딩(Multi-Threading)에서 동작한다. Snort는 싱글 스레딩(Single-Threading) 환경이어서 네트워크로부터 패킷을 수집하는 과정에서부터 위협을 탐지해내고 이를 바탕으로 경고를 발생시키는 과정까지 모두 하나의 CPU Core에서 처리된다. 반면, Suricata의 경우 멀티 쓰레딩 환경으로 여러 CPU Core에서 하나의 탐지 작업을 동시에 수행할 수 있도록 설계되어있다.

3. Snort와 Suricata의 비교

3.1 탐지 기능 비교

앞서 언급하였듯이 Suricata는 Snort를 대체하기 위한 침입탐지시스템으로서 개발되었다.

이렇듯 Suricata가 Snort의 대안으로서의 역할을 제대로 수행하기 위해서 기존의 Snort가 가지고 있었던 기능을 갖추고 있어야 할 필요가 있었다. 이러한 이유로 Suricata는 Snort와 호환이 가능하도록 실제로 Snort에서의 탐지 기능을 대부분 갖추고 있다.

그리고 동시에, 기존의 Snort에는 존재하지 않았던 여러 기능이 추가되기도 하였다.

추가된 기능 중에서 대표적으로 세 가지 정도를 소개하자면, Protocol Identification과 HTTP Normalizer & Parser, 그리고 File Identification 정도로 정리해볼 수 있다.

3.1.1 Protocol Identification

기존의 Snort에서는 IP, ICMP, UDP, TCP를 제외한 데이터 패킷의 프로토콜을 탐지해낼 때, 프로토콜마다 부여받은 고유의 포트 번호에 의존하곤 했다. 가령 예를 들면, Snort가 패킷 스니핑을 통해 받아들인 데이터 패킷에서 포트 번호가 80으로 나타난 경우, Snort는 이 데이터 패킷을 HTTP 패킷으로 인식하게 되는 식이었다.

그런데, Suricata에서는 앞서 언급한 네 가지 프로토콜 이외에 HTTP, FTP, TLS, SMB, DNS와 같이 자주 사용되는 응용 계층 프로토콜에 대해서도 포트 번호를 이용해 프로토콜을 탐지하는 것이 아니라 프로토콜 자체를 바로 탐지해낼 수 있도록 기능이 추가되었다. 이에 따라 Suricata에서는 사용자가 HTTP,

FTP, TLS, SMB, DNS에 대해서도 포트 번호에 의존하지 않고 프로토콜 자체에 대해서 규칙을 지정할 수 있도록 되어 있다.

다음은 124.50.4.0/24가 아닌 IP주소로부터 내부네트워크로 전달되는 ftp 패킷을 모두 버리도록 하는 명령이다. Suricata에서는 FTP의 고유 포트번호인 21을 지정하지 않고서도 FTP 패킷을 탐지해낼 수 있다.

```
drop ftp !124.50.4.0/24 any -> $HOME_NET a
ny (msg:"drop FTP Packet"; sid:1, rev:1)
```

3.1.2 HTTP Normalizer & Parser

Suricata는 HTTP Normalizer & Parser(HTP Library)를 갖춤으로써 좀 더 발전된 HTTP Stream 처리를 가능하게 했다. 또한 이에 따라 Suricata는 OSI 7 Layer Model 중에서 7계층에 해당하는 응용 계층의 네트워크 트래픽을 이해할 수 있게 되었다.

이에 대한 예시로 다음과 같이 규칙 설정을 하면, HTTP Payload에서 uri가 "/index.html"인 HTTP Request 패킷을 탐지해낼 수 있다.

```
alert tcp $HOME_NET any -> $EXTERNAL_
NET 80 (msg:"Detect HTTP Request whose ur
i is index.html"; content:"/index.html";http_ur
i sid:1; rev:1;)
```

3.1.3 File Identification & Extraction

File Identification은 단어 자체에서 의미하듯이 File을 인식하여 탐지할 수 있는 기능이 Suricata에 추가되었다. 이 기능을 이용하면 Suricata를 통해 네트워크를 지나다니는 여러 파일을 인식할 수 있고, 파일을 인식한 다음 이를 추출하여 디스크에 저장할 수도 있다.

File Identification과 File Extraction을 하기 위해서는 우선, Suricata의 설정파일(Configuration File)인 Suricata.yaml에서 "file-store" 옵션을 다음과 같이 설정하도록 해야 한다.

```
- file-store:
  enabled: yes #파일 추출 시 yes로 설정
  log-dir: files #파일 저장 디렉토리 위치
  force-magic: no
  force-md5: no
  waldo: file.waldo
```

다음으로, 규칙 설정을 통해서 File Identification과 Extraction이 가능하다. 가령 예를 들어, 확장자가 jpg 파일을 추출하여 디스크에 저장하기 위해서는 다음과 같이 규칙을 설정하여야 한다.

```
alert http any any -> any any (msg:"Extract J
PEG Image Files"; fileext:"jpg"; filestore; sid:
2; rev:1;)

# fileext:"String"은 인식하고자 하는 파일의 확
장자를 설정하는 것이고, filestore는 인식된 파일
을 추출하여 디스크에 저장할 때 쓰는 옵션이다.
```

3.2 양적 처리 성능 비교

3.2.1 멀티 스레딩(Multi-Threading) 설정

앞서 설명하였듯이 Suricata는 탐지 기능의 추가 이외에도 기존의 Snort에서의 싱글 스레딩(Single-Threading)에서 벗어나 새로운 처리 방식으로 멀티 스레딩(Multi-Threading)을 도입하였다.

이러한 변화는 Suricata로 하여금 단순히 기존의 Snort에 비해서 탐지해낼 수 있는 범위가 넓어지도록 한 것뿐만 아니라 양적으로 패킷을 처리해내는 속도 역시 변화할 수 있도록 하였다.

Suricata에서의 멀티 스레딩은 여러 CPU Core를 가진 컴퓨터 환경에서 이용되는 것으로, 패킷을 스니핑하여 수집하는 작업부터 규칙과 비교하는 탐지 작업 등 세부적인 작업들을 각각 하나의 스레드(Thread)로 나누어 각각의 스레드에 대하여 해당 스레드의 작업이 동시에 여러 CPU Core에서 이루어질 수 있도록 설정하는 방식으로 이루어진다.

이러한 설정은 Suricata의 설정 파일(Configuration

File)인 Suricata.yaml에서 가능하다. 다음 예시에서 볼 수 있듯이, 설정 파일인 Suricata.yaml에서는 침입 탐지 과정에서 세부적 작업 스레드마다 cpu라는 옵션을 통해 해당 작업 스레드에 몇 개의 CPU Core를 배분할 것인지 설정할 수 있다. 이에 비추어 다음 예시에 대해 살펴보면, 탐지 작업 스레드(Detecting Thread)에 해당하는 "-detect-cpu-set"에서 cpu 옵션을 "all"로 설정되어 있으므로 탐지 작업은 모든 CPU Core에서 이루어진다는 것을 알 수 있다.

```
cpu-affinity:
  - receive-cpu-set:
      cpu: [ 0 ]
  - decode-cpu-set:
      cpu: [ 0, 1 ]
      mode: "balanced"
  - stream-cpu-set:
      cpu: [ "0-1" ]
  - detect-cpu-set:
      cpu: [ "all" ]
      mode: "exclusive"
      prio:
        low: [ 0 ]
        medium: [ "1-2" ]
        high: [ 3 ]
        default: "medium"
  - output-cpu-set:
      cpu: [ "all" ]
      prio:
        default: "medium"
```

3.2.2 Snort와 Suricata의 처리 속도 비교

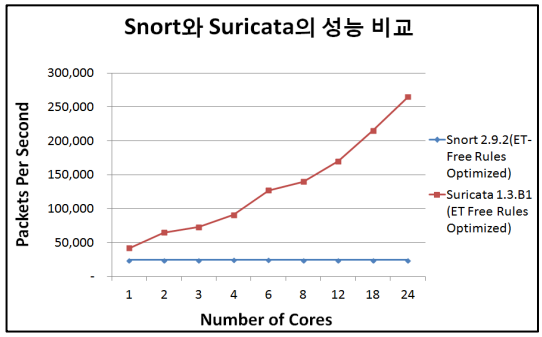
이렇듯 Suricata에서는 Snort와는 달리 특정 작업 스레드에 대해서 여러 CPU Core에서 동시에 처리될 수 있도록 사용자가 설정할 수 있다.

따라서 당연히 양적으로 처리 속도가 기존의 Snort와는 다를 것이라는 것을 짐작해볼 수 있다. 그리하여 본 연구에서는 Snort와 Suricata의 처리 속도에서의 차이가 나타나는지 확인해보고자 한다.

이와 관련해서 White, Fizsimmons 그리고 Matthews의 연구 "Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata"를 살펴보면, Suricata와 Snort의 처리 속도에서 다음과 같은 차이가 나타난다는 사실을 확인할 수 있었다.

이 연구에서는 침입탐지 작업에 사용되는 CPU Core의 개수를 독립변수(Independent Variable)로 설정하여, CPU Core의 개수에 따라 달라지는 종속변수(Dependent Variable)인 처리되는 초당 패킷의 개수(PPS, Packets Per Second)를 관찰하였다. 다만, 이 연구에서는 하나의 데이터 패킷에 대해서는 Packet Acquisition 작업부터 Detection, Output 작업까지 하나의 CPU Core에서 이루어지도록 했다.

그 결과, 다음과 같이 Snort의 경우에는 PPS의 값이 CPU Core의 개수와는 상관없이 Suricata의 PPS 수치보다 하회하는 수치를 보이고 있다. 반면, Suricata에서는 실험이 이루어진 모든 CPU Core 개수에서 Snort의 PPS 수치를 상회하고 있었고, 침입탐지 작업에 사용되는 CPU Core의 개수가 증가함에 따라 PPS 역시 높은 정적 상관관계(Positive Correlation)를 보여주면서 증가하는 것을 확인해볼 수 있다. [3]



(그림 2) Snort와 Suricata의 성능 비교

이러한 결과를 볼 때, Suricata의 패킷 처리 속도는 대체적으로 Snort에 비해 양적으로 월등하다는 사실을 파악할 수 있다. 더군다나, 오늘날과 같이 Multi Core가 보편화된 시점에서는 CPU Core의 개수가 늘어남에 따라 PPS 수치 역시 높아지는 Suricata가 Snort에 비해 더욱 월등한 처리 속도를 보일 수 있을 것으로 기대된다.

4. 결 론

지금까지 Snort와 Suricata에 대해서 탐지 기능 측면에서 어떤 차이가 있는지에 대해서 정리해보았고, Snort와 Suricata 사이에 처리 속도의 차이 역시 존재한다는 사실을 파악해볼 수 있었다.

Suricata는 본래 Snort의 대안으로서의 역할을 수행하기 위해서 개발되었다고 할 수 있는데, 본 논문에서 살펴본 바에 따르면 Suricata는 기존의 Snort에 비해서 탐지해낼 수 있는 패킷의 범위가 상대적으로 더 커졌고 양적인 측면에서의 처리 속도 역시 괄목할만하게 증가했음을 알 수 있다.

이런 점을 바탕으로 종합적으로 판단하여 볼 때, Suricata는 비록 아직까지는 Snort에 비해 덜 보편화되어 있다고는 하지만, 탐지 기능 측면에서의 발전과 처리 속도의 향상이라는 주요 특징을 고려해볼 때 기존의 Snort의 대안으로 적절할 것으로 보인다.

참고문헌

[1] Jay Beale, James C. Foster, Jeffrey Posluns, Brian Caswell, '스노트 2.0 마술상자', 에이콘, 2003.

[2] Albin Eugene, "A comparative analysis of the Snort and Suricata intrusion-detection systems", Master's thesis, NAVAL POSTGRADUATE SCHOOL, 2011.

[3] Joshua S. White, Thomas T. Fitzsimmons, Jeanna N. Matthews, "Quantitive Anaylsis of Intrusion Detection Systems: Snort and Suricata", PROCEEDINGS - SPIE THE INTERNATIONAL SOCIETY FOR O, Vol. 8757, 2013.

[4] 김윤정, "2계층 구조의 탐지 규칙을 사용하는 탐지 기법 및 SNORT에의 구현", 정보기술논문지, Vol. 5, pp. 79-85, 2007.

[5] 김윤정, 박유미, "침입탐지시스템의 탐지모듈 성능개선 방안에 대한 연구", 정보기술논문지, Vol. 1, pp. 1-8, 2003.

[6] 손형서, 이성운, 김현성, "Rule Protecting Scheme for Snort", 한국정보기술응용학회학술대회, Vol. 2005, No. 1, pp. 259-262, 2005.

[저 자 소 개]

정 명 기 (Myeongki Jeong)

성균관대학교 컴퓨터교육과 소속

email : uyh45@skku.edu

박 원 형 (Won Hyung Park)

2002년 서울과학기술대학교 산업정보시스템공학과 공학사 학사
2005년 서울과학기술대학교 정보산업공학과 공학석사 석사
2009년 경기대학교 정보보호학과 이학박사 박사

email : whpark@kdu.ac.kr

안 성 진 (Seongjin Ahn)

1988년 2월 성균관대학교 정보공학과 (공학사)학사
1990년 2월 성균관대학교 대학원 정보공학과 공학석사 석사
1998년 8월 성균관대학교 대학원 정보공학과 공학박사 박사

email : sjahn@skku.edu