Docker 컨테이너 활용
Snort 운영

2022.08

목 차

개요

1. 목적

이번 실습의 목적은 Docker 컨테이너를 활용하여 공격자는 플러딩 공격을 실시하고, 희생자는 snort 를 설치 및 운영하여 공격을 탐지하는 것에 있다.

2. 구성

- 이 실습은 Docker 환경에서 우분투와 칼리리눅스 컨테이너를 실행하여 진행하였다.
- 공격자는 Hping3 명령어를 이용하여 플러딩 공격을 실시한다.
- 플러딩 공격: 시스템에 과도한 부하를 일으켜 정보 시스템의 사용을 방해하는 공격 방식이다. 이번 실습에서는 아래 표의 3가지 플러딩 공격을 진행한다.

플러딩 공격	내용
TCP Syn	TCP 연결 특성을 이용한 Dos 공격의 일종으로 Unreachable 한 주소로 IP 를 위조하여 syn 을 보내 대상시스템이 더 이상의 연결 요청을 받아드릴 수 없도록 만드는 공격
UDP	대량의 UDP 패킷을 이용하여 대상 호스트 네트워크 bandwidth 를 소모시켜 부하를 발생시키는 공격
ICMP	대량의 ICMP 패킷을 전송하여 ICMP echo reply 패켓을 발생시켜 부하를 발생시키는 공격

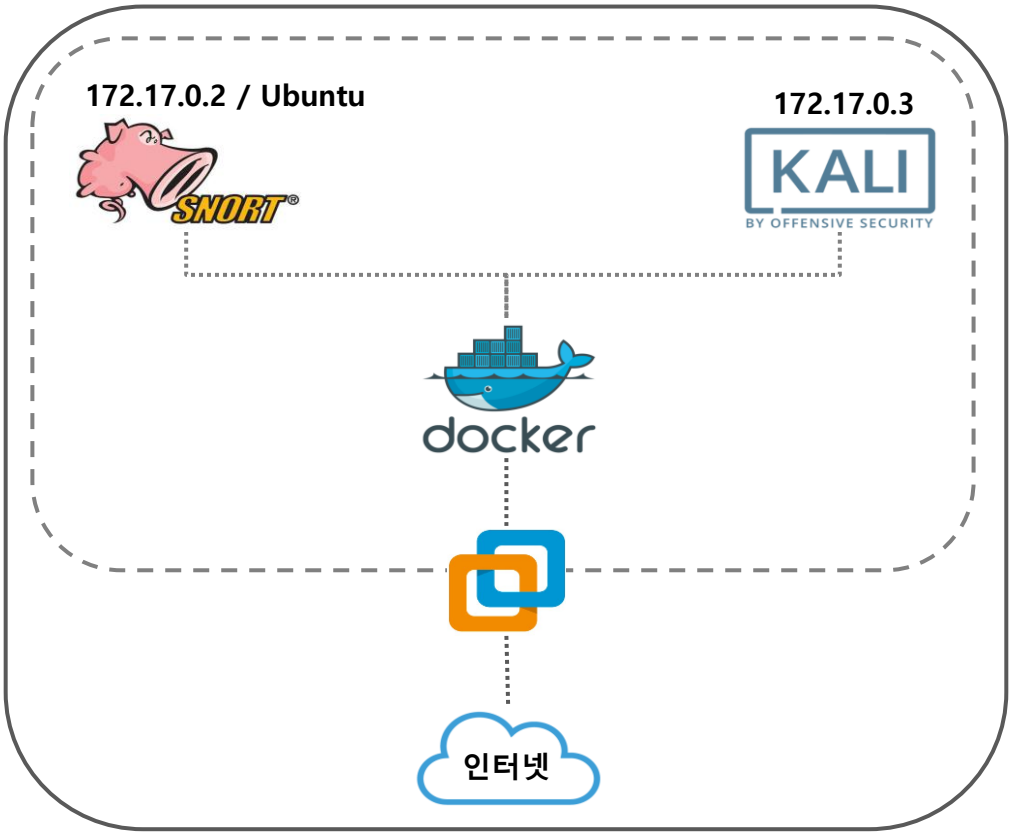
[표 1-1] 플러딩 공격의 종류와 설명

- Hping3: Hping3 명령어는 일반 ping 명령어와는 다르게 ICMP 패킷을 포함한 TCP/UDP/ICMP 등의 패킷 전송이 가능하다. 실습에서는 3.0.0-alpha-2 버전을 사용하여 공격을 진행하였다.
- 희생자는 Snort 를 이용하여 공격을 탐지한다.

✱ 참고: AhnLab 보안 용어사전

환경 구축

1. 네트워크 구성도



[그림 2-1] 네트워크 구성도

■ 사용 환경

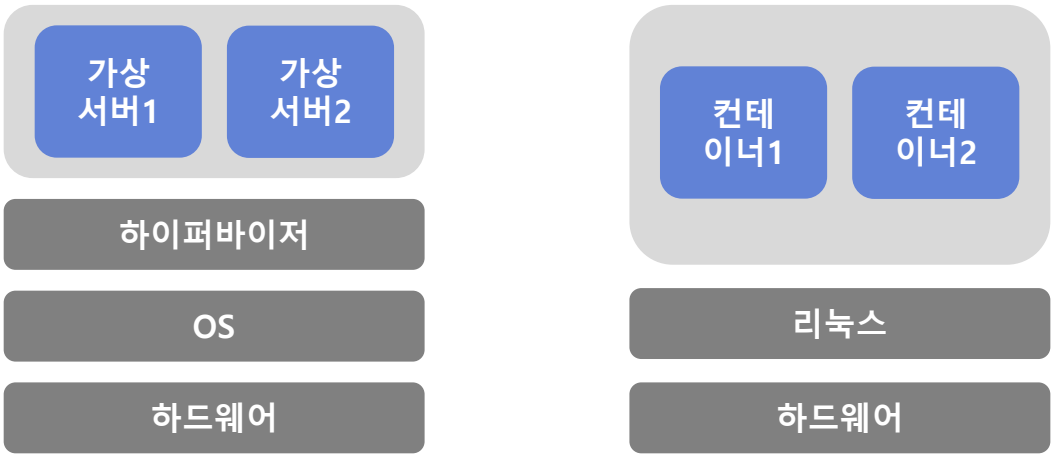
- VMware 16 Pro → 실습에서 사용할 하이퍼바이저 가상 환경
- Rocky Linux 9.0 → Docker 를 실행할 운영체제
- Docker 20.10.17 → 실습에서 사용할 컨테이너 가상 환경 플랫폼
- Ubuntu → 희생자 / Snort 설치 / Docker 컨테이너 / IP:172.17.0.2
- Kali Linux → 공격자 / Docker 컨테이너 / IP:172.17.0.3

2. 환경 설정

■ 하이퍼바이저 / 컨테이너 가상화

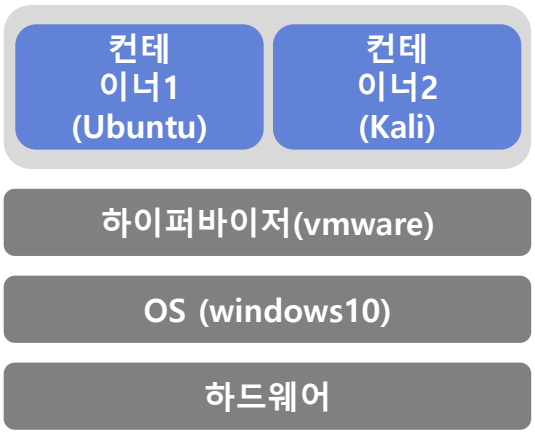
- VMware 나 Virtualboc 처럼 호스트 PC에 가상소프트웨어를 설치하여 가상 환경을 구축하는 기법을 하이퍼바이저 가상화라고 한다.

- 하이퍼바이저 가상화보다 더 적은 리소스로 가상 환경을 구축하는 기법을 컨테이너 가상화라고 한다.



[그림 2-2] 하이퍼바이저 가상화(왼쪽) 와 컨테이너 가상화(오른쪽)

- 이번 실습에서는 하이퍼바이저 가상화 환경에서 컨테이너 가상화 환경을 구축하였다.
  → windows10 에서 vmware 를 이용하여 Docker 를 사용 하였다.



[그림 2-3] 하이퍼바이저 와 컨테이너 가상화 환경을 혼합한 실습 환경

Docker

1. Docker의 정의

■ Docker / Docker 이미지 / Docker 컨테이너

- Docker: 대표적인 컨테이너 가상화 플랫폼
- Docker 이미지: 컨테이너를 구동하기 위한 읽기 전용 템플릿
- Docker 컨테이너: 이미지를 이용하여 실행된 독립적인 프로세스

Docker	이미지	컨테이너
• 하나의 PC에서 여러대의 가상 PC를 운영할 수 있도록 한다. • 소프트웨어적으로 물리적인 환경을 구성한다. • 컨테이너 가상화 타입의 리눅스 기반 오픈소스 플랫폼이다.	• 컨테이너를 구동하기 위한 실행 파일과 설정 파일의 세트이다. • 직접 제작도 가능하며, Docker 허브에서 다운로드를 받을 수 있다.	• 읽기/쓰기가 가능하며 생성된 데이터는 볼륨에 저장한다. • 실행된 컨테이너는 IP 주소가 할당되며 마치 하나의 독립적인 서버처럼 동작한다.

[표 3-1] Docker, 이미지, 컨테이너 설명

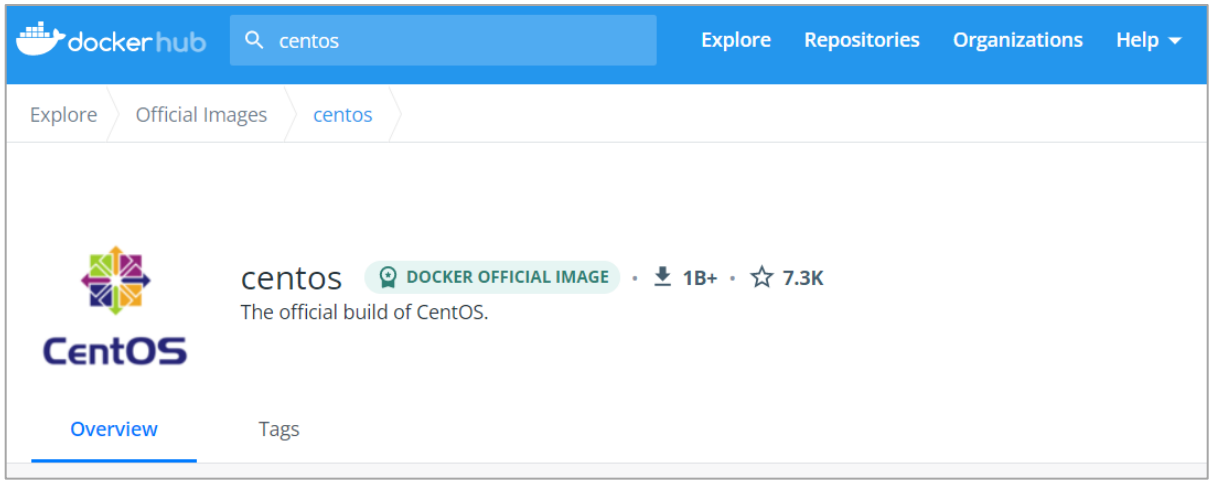
■ Docker 특징

- 리소스 소모가 적다.
  → 필요한 애플리케이션만 구동할 수 있도록 이미지를 만들기 때문에 리소스가 적다.

```
[root@Rocky9 ~]# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
openwrtorg/rootfs         latest    a4c66acaa0ff   4 days ago       9.53MB
kalilinux/kali-rolling    latest    1a3b9c53e634   6 days ago       121MB
ubuntu                    latest    df5de72bdb3b   3 weeks ago      77.8MB
hello-world               latest    feb5d9fea6a5   11 months ago    13.3kB
ciscotalos/snort3         latest    2c50bee6393c   2 years ago      774MB
vulnerables/web-dvwa      latest    ab0d83586b6e   3 years ago      712MB
```

[그림 3-1] Docker 이미지 목록

- 이미지 공유가 쉽다.
  → 명령어 만으로 Docker 허브에서 이미지를 공유 할 수 있다. 또한, 사설로 저장소를 운영할 수 있기 때문에 보안적인 측면에서 걱정을 덜 수 있다.



[그림 3-2] Docker hub 에서 Centos 공식 이미지 검색

- 새로운 이미지를 만들기 쉽다.
  → Dockerfile을 이용하여 손쉽게 이미지를 생성할 수 있다.

```
# Docker Images
FROM centos

# Author
MAINTAINER Yongrack Son

# 사용자 추가 및 사용자 확인
RUN ["useradd", "user1"]
RUN ["whoami"]
RUN ["id"]

# 사용자 확인
USER Test
RUN ["whoami"]
RUN ["id"]
```

[그림 3-3] Dockerfile 을 이용한 이미지 생성

2. Docker 설치

■ Rocky Linux 에 Docker 설치

- Docker를 다운받기 위해 Docker 저장소를 Rocky Linux 에 추가

```
~# yum install -y yum-utils

~# yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
```

- Docker 설치파일(Docker-ce) 설치

```
~# yum install -y docker-ce
```

- Docker 설치 확인

```
~# rpm -qa | grep docker
```

- 정상적으로 Docker가 설치 되었다면 아래와 같이 출력된다.

```
[root@Rocky9 ~]# rpm -qa | grep docker
docker-scan-plugin-0.17.0-3.el9.x86_64
docker-ce-cli-20.10.17-3.el9.x86_64
docker-ce-rootless-extras-20.10.17-3.el9.x86_64
docker-ce-20.10.17-3.el9.x86_64
docker-compose-plugin-2.6.0-3.el9.x86_64
```

[그림 3-4] Docker 설치 완료

- Docker 서비스 시작

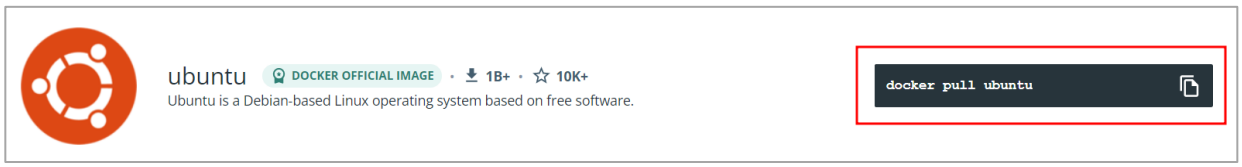
```
~# systemctl enable --now docker.service

~# systemctl status docker.service
```

3. Docker 기본 명령어

■ Docker hub 에서 이미지 다운로드

```
~# docker pull ubuntu                    #ubuntu 컨테이너 다운로드
```

- docker pull [이미지 이름] 을 입력하면 docker hub 로부터 이미지를 다운 받는다.
- docker hub 사이트에서도 해당 명령어를 복사하여 이미지를 다운 받을 수 있다.



[그림 3-5] docker hub 사이트 이미지 다운로드 명령어 복사 배너

■ Docker images 목록 출력

```
~# docker images
```

- [그림 3-6] 과 같이 Repository 에는 이미지 이름, Tag 에는 이미지의 tag, Image ID 에는 이미지의 ID, Created 에는 이미지 생성시기, Size 에는 이미지 크기가 표시된다.

```
[root@Rocky9 ~]# docker images
REPOSITORY              TAG       IMAGE ID       CREATED       SIZE
kalilinux/kali-rolling  latest    1a3b9c53e634   8 days ago    121MB
```

[그림 3-6] docker images 입력 후 출력된 결과

■ 컨테이너 실행

```
~# docker container run --name testubuntu --rm -it -p 8080:80 ubuntu
```

- docker run 또는 docker container run 을 입력하면 컨테이너를 실행시킨다.
- 컨테이너 실행 시 미리 다운 받아 놓은 이미지가 없다면, 자동으로 docker hub 에서 다운받아 실행시킨다.
- 위 명령어에서 사용된 옵션에 대한 설명은 아래 표와 같다.

옵션	내용
--rm	컨테이너를 종료되면, 해당 컨테이너가 자동으로 삭제된다.
--name	컨테이너의 이름을 지정한다.
-i	컨테이너의 표준 입력을 연다. 보통 -t 와 함께 사용한다.
-t	tty 를 사용한다. -i 와 함께 사용하면, 컨테이너 내부로 접근하여 shell 을 실행하거나 명령어를 실행할 수 있다.
-p	포트를 지정하는 옵션이다.

[표 3-2] docker run 옵션 설명

- 위 명령어를 입력하여 출력되는 결과는 아래의 [그림 3-7] 과 같다.

```
[root@Rocky9 ~]# docker container run --name testubuntu --rm -it -p 8080:80 ubuntu
root@edd06141d00e:/#
```

[그림 3-7] docker run 명령어 입력 후 출력된 결과

Snort

1. Snort의 정의

■ Snort(Snort)

- 오픈소스 기반 대표적 IDS

- 네트워크 트래픽 감시 및 분석 프로그램

- 다양한 운영체제에서 설치, 실행 가능

■ IDS / IPS

구분	IDS	IPS
구축 방식	- 미러링 방식으로 패킷이 통과한 후 검사	- 인라인 방식으로 통과하는 패킷을 검사 한 후 전송
성격	- 패턴 등록 후 반응	- 공격 전 사전 차단
목적	- 침입 탐지	- 침입 방지 - 침입 탐지 후 적극 대응
부하	- 네트워크 부하 덜함	- 네트워크 부하 발생 가능
대응 방법	- 관리자에게 경고 알림	- 자동 차단
특징	- 오탐, 미탐 문제 발생 가능 → 지속적인 모니터링 필요	- 실시간으로 공격 패킷 차단 가능

[표 4-1] IDS 와 IPS 비교

✱참고: 빡통들을 위한 쉽고 재미있는 세상 모든 IT+ 정보보안
https://webstone.tistory.com/92

2. Snort 룰 구조

■ Snort 룰 (헤더 + 옵션)

header							option
action	**protocol**	**IP**	**port**	**->**	**IP**	**port**	**option**
①	②	③	④	⑤	⑥	⑦	⑧

[표 4-2] Snort 룰 구조

- [표 4-2]에 대한 Snort 룰 설명은 아래와 같다.

① action: 패킷 처리 방법을 지정
- alert: 경고를 발생시키고 로그에 기록
- log: 로그에 기록
- pass: 패킷을 무시
- drop: iptable 을 통해 패킷 차단 후 로그에 기록

② 패킷의 프로토콜 종류를 지정
- tcp / udp / icmp

③⑥ 송수신자의 IP 주소 지정
- any: 임의의 모든 IP

④⑦ 송신자 포트 지정
- any: 임의의 모든 포트

⑤ 방향: 패킷의 방향을 나타내는 기호 지정

⑧ 옵션
- msg: 지정한 메세지가 alert 발생시 이벤트 이름으로 사용
- flags: TCP 플래그비트가 있는지 확인
- threshold: 대량의 패킷에 대응하기 위한 옵션 (type / tracck / count / seconds)
- sid: 시그니처 ID 지정

플러딩 공격 및 Snort 탐지

1. Docker 이미지 다운로드 및 컨테이너 실행

실습은 vmware 16 / Rocky Linux 9.0 환경에서 실행한다.

■ Docker 허브에서 이미지 다운로드

- Ubuntu Docker 허브 이미지 URL: https://hub.docker.com/_/ubuntu

- Kali Linux: Docker 허브 이미지 URL: https//hub.docker.com/r/kalilinux/kali-rolling

```
~# docker pull ubuntu                          # ubuntu 설치
~# docker pull kalilinux/kali-rolling          # kali 설치
~# docker images                               # 이미지 설치 확인
```

- 이미지가 제대로 설치되었는지 확인한다.

```
[root@Rocky9 ~]# docker images
REPOSITORY               TAG       IMAGE ID       CREATED       SIZE
kalilinux/kali-rolling   latest    1a3b9c53e634   7 days ago    121MB
ubuntu                   latest    df5de72bdb3b   3 weeks ago   77.8MB
```

[그림 5-1] 이미지 다운로드 완료 확인

■ 컨테이너 생성

```
~# docker run -it ubuntu                       # ubuntu 컨테이너 실행
~# docker run -it  kalilinux/kali-rolling      # kali 컨테이너 실행
~# docker container ls -a                      # 컨테이너 확인
```

- Ubuntu 컨테이너 실행 후 Ctrl + p + q 를 입력하면 컨테이너를 종료 시키지 않고 빠져 나올 수 있다.

```
[root@Rocky9 ~]# docker container ls -a
CONTAINER ID   IMAGE                    COMMAND       CREATED      STATUS                      PORTS     NAMES
b8751c2169ed   vulnerables/web-dvwa     "/main.sh"    3 days ago   Exited (137) 3 days ago               dvwa
9e737bc034d0   ubuntu                   "bash"        4 days ago   Up 45 seconds                         ubuntu
a5553ae6c489   kalilinux/kali-rolling   "/bin/bash"   4 days ago   Up 12 seconds                         kali1
```

[그림 5-2] 컨테이너 생성 완료 확인

2. IP 주소 확인

■ Ubuntu IP 주소 확인

```
~# ifconfig                                    # 172.17.0.2
```

```
root@9e737bc034d0:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 37  bytes 5334 (5.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

[그림 5-3] Ubuntu에서 IP 주소 확인

■ Kali Linux IP 주소 확인

```
~# ifconfig                                    # 172.17.0.3
```

```
┌──(root㉿a5553ae6c489)-[/]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.3  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:03  txqueuelen 0  (Ethernet)
        RX packets 16  bytes 2092 (2.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

[그림 5-4] Kali Linux 에서 IP 주소 확인

3. Ubuntu 에 Snort 설치

■ Snort 설치

```
~# apt-get update                                          # update 실행
~# apt-get install -y zlib1g-dev liblzma-dev               # snort 패키지 설치
~# apt-get install -y libnghttp2-dev openssl libssl-dev    # snort 패키지 설치
~# apt-get install -y snort                                # snort 설치
```

- Snort 환경에 필요한 관련 패키지들을 먼저 설치한 후 snort 를 다운받는다.

■ Snort 버전 확인

```
~# snort -v                                    # snort 버전 확인
```

- 정상적으로 설치가 되었는지 확인하기 위해 버전을 확인한다.

```
root@9e737bc034d0:/# snort -v
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15.1 GRE (Build 15125)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11
```

[그림 5-5] snort 설치 후 버전 확인

■ Snort 구성

```
~# vi /etc/snort/snort.conf                    # snort 설정파일 수정
```

- 네트워크 주소를 변경한다. → HOME_NET 을 172.17.0.2(ubuntu IP) 로 설정

```
60 # Note to Debian users: this value is overriden when starting
61 # up the Snort daemon through the init.d script by the
62 # value of DEBIAN_SNORT_HOME_NET s defined in the
63 # /etc/snort/snort.debian.conf configuration file
64 #
65 ipvar HOME_NET 172.17.0.2
```

[그림 5-6] 네트워크 주소 변경

- 적용시킬 룰을 저장하는 파일 설정 → 실습에서는 local.rules 에 룰을 저장 시킬 것이므로 이 규칙을 제외한 나머지 규칙은 주석처리 한다.

```
589 # site specific rules
590 include $RULE_PATH/local.rules
591
592 # The include files commented below have been disabled
593 # because they are not available in the stock Debian
594 # rules. If you install the Sourcefire VRT please make
595 # sure you re-enable them again:
596
597 ##include $RULE_PATH/app-detect.rules
598 ##include $RULE_PATH/attack-responses.rules
599 ##include $RULE_PATH/backdoor.rules
600 ##include $RULE_PATH/bad-traffic.rules
601 ##include $RULE_PATH/blacklist.rules
602 ##include $RULE_PATH/botnet-cnc.rules
603 ##include $RULE_PATH/browser-chrome.rules
604 ##include $RULE_PATH/browser-firefox.rules
605 ##include $RULE_PATH/browser-ie.rules
606 ##include $RULE_PATH/browser-other.rules
```

[그림 5-7] 룰을 저장할 파일 설정

4. Snort 룰 설정

■ local.rules 파일에 룰 작성

```
~# vi /etc/snort/rules/local.rules
```

- 실행할 공격 기법들을 탐지하기 위한 룰을 local.rules 파일에 작성하고 저장한다.

```
# SYN Flooding
alert tcp any any -> 172.17.0.2 80 (msg:"SYN-Flooding-Detection"; flags: S; threshold: type
threshold, track by_dst, count 10, seconds 1; sid: 1000004;)

# UDP Flooding
alert udp any any -> 172.17.0.2 any (msg:"UDP-Flooding-Detection"; threshold: type threshold
, track by_dst, count 10, seconds 1; sid: 1000002;)

# ICMP Flooding
alert icmp any any -> 172.17.0.2 any (msg:"ICMP-Flooding-Detection"; threshold: type thresho
ld, track by_dst, count 10, seconds 1; sid: 1000003;)
```

[그림 5-8] Snort 룰 작성 및 저장

- [그림 5-8] 에서 처럼 작성한 snort 룰에 대한 설명은 아래와 같다.

TCP Syn Flooding
→ 172.17.0.2의 80포트로 접근하는 tcp 패킷 탐지한다. "SYN-Flooding-Detection" 이라는 메세지를 출력하며 도착 IP 주소에 기반해 추적하고 패킷 개수에 기반해 1초 동안 5회 이상 탐지가 되면 이벤트를 출력한다.

UDP Flooding
→ 172.17.0.2 으로 접근하는 udp 패킷을 탐지한다. "UDP-Flooding-Detection" 이라는 메세지를 출력하며 도착 IP 주소 및 패킷 개수에 기반해 추적, 1초 동안 10회 이상 탐지가 되면 이벤트를 출력한다.

ICMP Flooding
→ 172.17.0.2 으로 접근하는 icmp 패킷을 탐지한다. "ICMP-Flooding-Detection" 라는 메세지를 출력하며 도착 IP 주소 및 패킷 개수에 기반해 추적, 1초 동안 10회 이상 탐지가 되면 이벤트를 출력한다.

5. Snort 실행

■ 설정한 룰 파일과 함께 snort 를 실행시킨다.

```
~# snort –c /etc/snort/rules/local.rules
```

- snort가 정상적으로 실행되면 [그림 5-9] 와 같은 화면이 출력된다.

```
+----------------------[event-filter-global]---------------------------------
| none
+----------------------[event-filter-local]----------------------------------
| gen-id=1      sig-id=1000003    type=Threshold tracking=dst count=10  seconds=1
| gen-id=1      sig-id=1000002    type=Threshold tracking=dst count=10  seconds=1
| gen-id=1      sig-id=1000004    type=Threshold tracking=dst count=5   seconds=10
+----------------------[suppression]-----------------------------------------
| none
--------------------------------------------------------------------------------
Rule application order: pass->drop->sdrop->reject->alert->log
Verifying Preprocessor Configurations!

[ Port Based Pattern Matching Memory ]
Unable to open directory "/usr/lib/daq"
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Reload thread starting...
Reload thread started, thread 0x7fed073a5640 (13)
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15.1 GRE (Build 15125)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.11

Commencing packet processing (pid=12)
```

[그림 5-9] Snort 실행 화면

6. Kali 공격 준비

■ hping3 설치

```
~# apt install hping3                                    # hping3 설치
```

- hping3 가 설치되어 있지 않기 때문에 명령어를 통해 설치한다.

■ hping3 사용법

- 명령어 구조:

```
~# hping3 <attack mode> <options> <source> <destination> <port> <count>
```

- 주요 옵션

옵션	설명
-h	도움말
-c	패킷을 몇 개 보낼지 결정
-i	패킷을 보낼 때 속도 결정
-n	숫자 추가
-p	목적지 포트
-s	출발지 포트
-S	TCP Syn
--flood	대량의 패킷 전송
--rand-dest	랜덤한 목적지 IP
--rand-source	랜덤한 출발지 IP
--fast	1초에 10개의 패킷
--faster	1초에 100개의 패킷

[표 5-1] hping3 주요 옵션 설명

7. TCP Syn 플러딩 공격

■ kali linux 에서 syn 패킷 전송

```
~# hping3 172.17.0.2  -S --rand-source -p 80 --fast
```

- hping3 설명: TCP Syn 패킷을 출발지 IP 를 랜덤하게 172.17.0.2의 80포트로 1초에 10개의 속도로 전송

■ Snort 로그 파일 확인

```
~# cat /var/log/snort/alert
```

- 탐지 기록이 텍스트로 저장되어 있는 alert 파일을 cat 명령어로 확인한다.

- 적용한 룰에 맞게 탐지되는 것을 확인할 수 있다.

```
[**] [1:1000004:0] SYN-Flooding-Detection [**]
[Priority: 0]
08/30-09:23:18.697943 143.106.124.190:10806 -> 172.17.0.2:80
TCP TTL:64 TOS:0x0 ID:6309 IpLen:20 DgmLen:40
******S* Seq: 0xC4EB36E  Ack: 0x47132473  Win: 0x200  TcpLen: 20

[**] [1:1000004:0] SYN-Flooding-Detection [**]
[Priority: 0]
08/30-09:23:18.698377 128.76.161.22:10811 -> 172.17.0.2:80
TCP TTL:64 TOS:0x0 ID:26229 IpLen:20 DgmLen:40
******S* Seq: 0x7C8837C1  Ack: 0x2046822D  Win: 0x200  TcpLen: 20

[**] [1:1000004:0] SYN-Flooding-Detection [**]
[Priority: 0]
08/30-09:23:18.698832 173.91.192.139:10816 -> 172.17.0.2:80
TCP TTL:64 TOS:0x0 ID:29592 IpLen:20 DgmLen:40
******S* Seq: 0x3C996FF6  Ack: 0x2C2E93F7  Win: 0x200  TcpLen: 20
```

[그림 5-10] /var/log/snort/alert 파일 중 TCP Syn 플러딩 탐지

8. UDP 플러딩 공격

■ kali linux 에서 udp 패킷 전송

```
~# hping3 172.17.0.2 --udp --rand-source -i u50 -c 100
```

- hping3 설명: UDP 패킷을 출발지 IP 를 랜덤하게 172.17.0.2로 1초에 50개의 속도로 100개의 패킷만 전송

■ Snort 로그 파일 확인

```
~# cat /var/log/snort/alert
```

- hping3 로 총 100개의 패킷을 보냈기 때문에, 총 10개의 alert 로그를 확인할 수 있다.

```
[**] [1:1000002:0] UDP-Flooding-Detection [**]
[Priority: 0]
08/30-09:33:25.370540 89.195.207.206:1229 -> 172.17.0.2:0
UDP TTL:64 TOS:0x0 ID:5793 IpLen:20 DgmLen:28
Len: 0

[**] [1:1000002:0] UDP-Flooding-Detection [**]
[Priority: 0]
08/30-09:33:25.372618 153.58.39.16:1239 -> 172.17.0.2:0
UDP TTL:64 TOS:0x0 ID:26556 IpLen:20 DgmLen:28
Len: 0

[**] [1:1000002:0] UDP-Flooding-Detection [**]
[Priority: 0]
08/30-09:33:25.374324 146.49.201.234:1249 -> 172.17.0.2:0
UDP TTL:64 TOS:0x0 ID:3019 IpLen:20 DgmLen:28
Len: 0
```

[그림 5-11] /var/log/snort/alert 파일 중 UDP 플러딩 탐지

9. ICMP 플러딩 공격

■ kali linux 에서 icmp 패킷 전송

```
# hping3 172.17.0.2 --icmp --rand-source -i u50 -c 100
```

- hping3 설명: ICMP 패킷을 출발지 IP 를 랜덤하게 172.17.0.2로 1초에 50개의 속도로 100개의 패킷만 전송

■ Snort 로그 파일 확인

```
~# cat /var/log/snort/alert
```

- hping3 로 출발지를 랜덤하게 설정했기 때문에 출발지가 alert 마다 다른 것을 확인 할 수 있다.

```
[**] [1:1000003:0] ICMP-Flooding-Detection [**]
[Priority: 0]
08/30-09:43:38.724735 242.180.213.104 -> 172.17.0.2
ICMP TTL:64 TOS:0x0 ID:65317 IpLen:20 DgmLen:28
Type:8  Code:0  ID:3328   Seq:4864  ECHO

[**] [1:1000003:0] ICMP-Flooding-Detection [**]
[Priority: 0]
08/30-09:43:38.726641 94.17.197.15 -> 172.17.0.2
ICMP TTL:64 TOS:0x0 ID:60705 IpLen:20 DgmLen:28
Type:8  Code:0  ID:3328   Seq:7424  ECHO

[**] [1:1000003:0] ICMP-Flooding-Detection [**]
[Priority: 0]
08/30-09:43:38.728403 29.238.245.30 -> 172.17.0.2
ICMP TTL:64 TOS:0x0 ID:37795 IpLen:20 DgmLen:28
Type:8  Code:0  ID:3328   Seq:9984  ECHO
```

[그림 5-12] /var/log/snort/alert 파일 중 ICMP 플러딩 탐지

컨테이너 업로드 및 저장

1. 컨테이너를 이미지로 생성하여 Docker hub에 업로드

- 사용했던 컨테이너를 이미지로 만들어 docker hub 에 업로드 하면, 다른 pc 에서도 Docker를 사용하여 컨테이너를 다시 다운 받아 사용할 수 있다.

■ 사용했던 컨테이너 이름 확인

```
~# docker container ls -a
```

```
[root@Rocky9 ~]# docker container ls -a
CONTAINER ID   IMAGE                  COMMAND       CREATED       STATUS                      PORTS     NAMES
b8751c2169ed   vulnerables/web-dvwa   "/main.sh"    5 days ago    Exited (137) 5 days ago               dvwa
9e737bc034d0   ubuntu                 "bash"        6 days ago    Exited (0) 12 minutes ago             ubuntu
a5553ae6c489   kalilinux/kali-rolling "/bin/bash"   6 days ago    Exited (137) 24 hours ago             kali1
```

[그림 6-1] Docker 컨테이너 이름 확인

■ commit 명령어를 이용하여 사용했던 컨테이너를 이미지로 제작

```
~# docker container commit -a "yong rack son" -m "ubuntu/snort-img" ubuntu
   sohn0322/testubuntu:snort
```

- 만든이: yong rack son, 이미지이름: sohn0322/testubuntu, tag: snort 로 이미지 제작

```
[root@Rocky9 ~]# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
sohn0322/testubuntu       snort     165ec800aef6   55 seconds ago   803MB
kalilinux/kali-rolling    latest    1a3b9c53e634   9 days ago       121MB
ubuntu                    latest    df5de72bdb3b   4 weeks ago      77.8MB
```

[그림 6-2] 컨테이너를 이미지로 생성 완료

■ Docker hub 로그인

```
~# docker login
```

```
[root@Rocky9 ~]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https
://hub.docker.com to create one.
Username: sohn0322
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

[그림 6-3] Docker 허브 로그인 완료

■ Docker hub에 이미지 업로드

```
~# docker image push sohn0322/testubuntu:snort
```

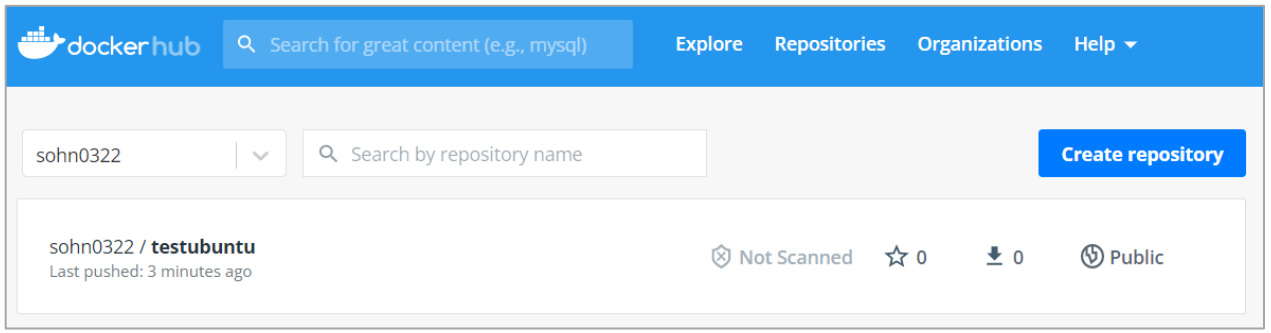
- sohn0322/testubuntu:snort 는 '생성했던 이미지 이름:tag' 이다.

- 업로드가 성공적으로 완료 되었으면 [그림 6-1] 같은 결과가 출력된다.

```
[root@Rocky9 ~]# docker image push sohn0322/testubuntu:snort
The push refers to repository [docker.io/sohn0322/testubuntu]
8821309c2b98: Pushed
629d9dbab5ed: Mounted from library/ubuntu
snort: digest: sha256:8206dd2ef73ae43d1d07337ae72160c2093a9e8726ab8fac25b48b87f38f4059 size: 742
```

[그림 6-4] Docker hub 에 이미지 업로드 완료

- 이미지 업로드가 완료되면, Docker hub 에 접속하여 확인한다.



[그림 6-5] Docker hub 에서 확인한 업로드 된 이미지

2. 컨테이너를 tar 파일로 저장

- 사용했던 컨테이너를 tar 파일로 저장하면 인터넷이 안되는 환경에서도 적은 용량으로 저장했던 컨테이너를 사용할 수 있다.

■ 컨테이너를 tar 파일로 저장

```
~# docker container export kali1 > kali1.tar
```

- export 명령어를 이용하여 kali1 컨테이너를 tar 파일로 생성하고 확인한다.

```
[root@Rocky9 /test]# ls -l
합계 1273840
-rw-r--r--. 1 root root 1304408064 2022-08-31 21:20 kali1.tar
```

[그림 6-6] 사용했던 kali linux 컨테이너를 export 명령어를 이용해 tar 파일 생성

■ tar 파일을 이미지 파일로 생성

- import 명령어를 이용하면 tar 파일을 이미지로 생성할 수 있다.

```
[root@Rocky9 /test]# docker image import kali1.tar
sha256:9f1a891351fb98e1703cb80c104ed55cb26286e783bb0ac1521f5d92fb72b8be
[root@Rocky9 /test]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
<none>       <none>    9f1a891351fb   7 seconds ago    1.29GB
```

[그림 6-7] tar 파일을 이용하여 docker 이미지 생성

끝