

IPS Integration Guide to 2STON SPNBox

06.11.2019 ~

Doc. Revision: 0.2

Chunghan.Yi(michael@2ipco.com)
2IP R&D Center

Contents

Part I.

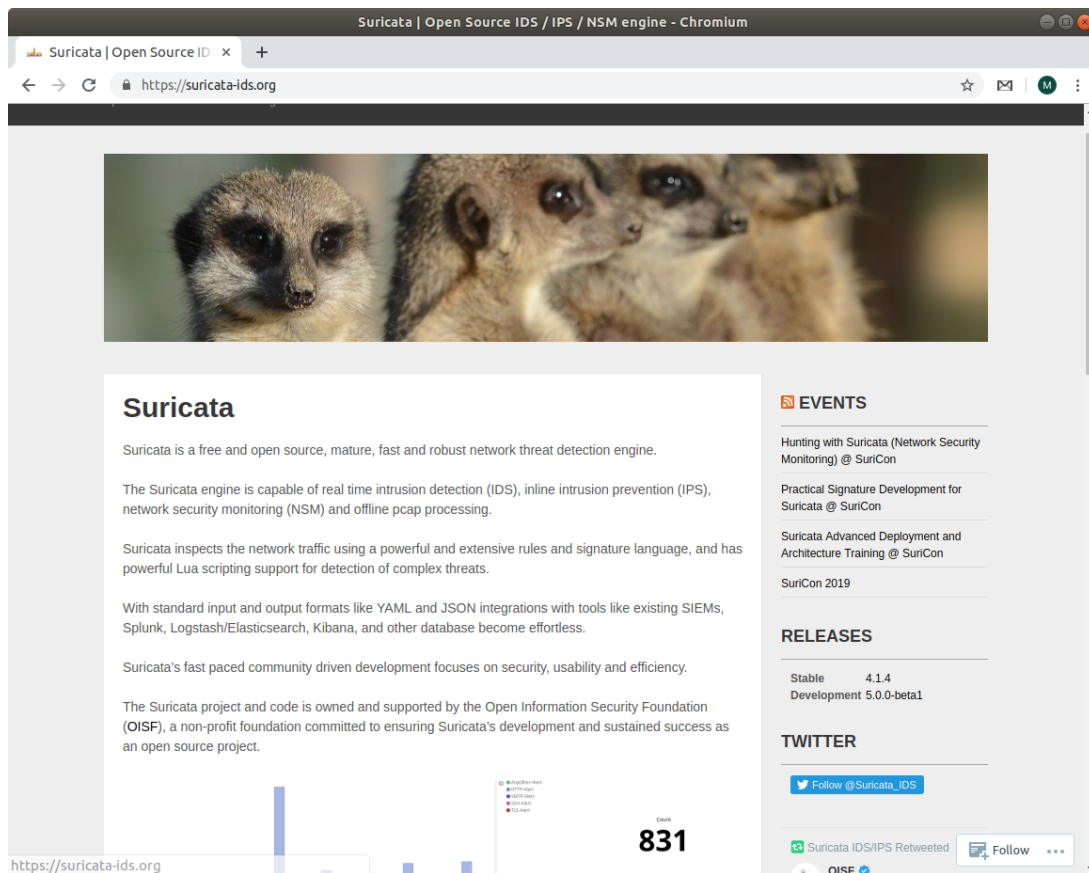
1. Suricata IDS/IPS Overview
2. How to build Suricata
3. How to run Suricata on SPNBox
4. Suricata IoT Source Cross Compile 하기
5. Suricata Rule Update[TBD]

Part II.

6. Snort Overview
7. How to build & run Snort on SPNBox
8. How to build Snort on OpenWrt
9. Snort Rule Update[TBD]

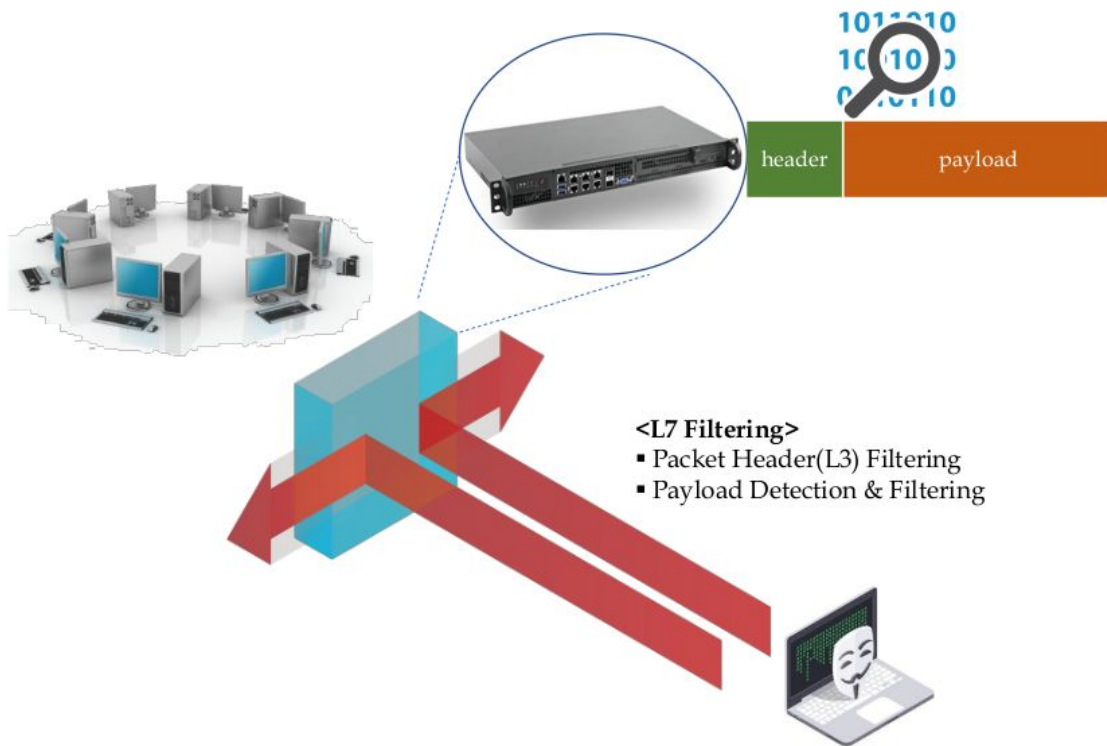
1. Suricata IDS/IPS Overview

IDS(Intrusion Detection System - 침입탐지 시스템) or IPS(Intrusion Prevention System - 침입방지 시스템)은 Firewall/VPN과 더불어 네트워크 보안의 중심을 이루는 기술이다. 이번 절에서는 Open source IDS/IPS로 유명한 snort 다음으로 등장한 **suricata**에 대해 소개하고자 한다.



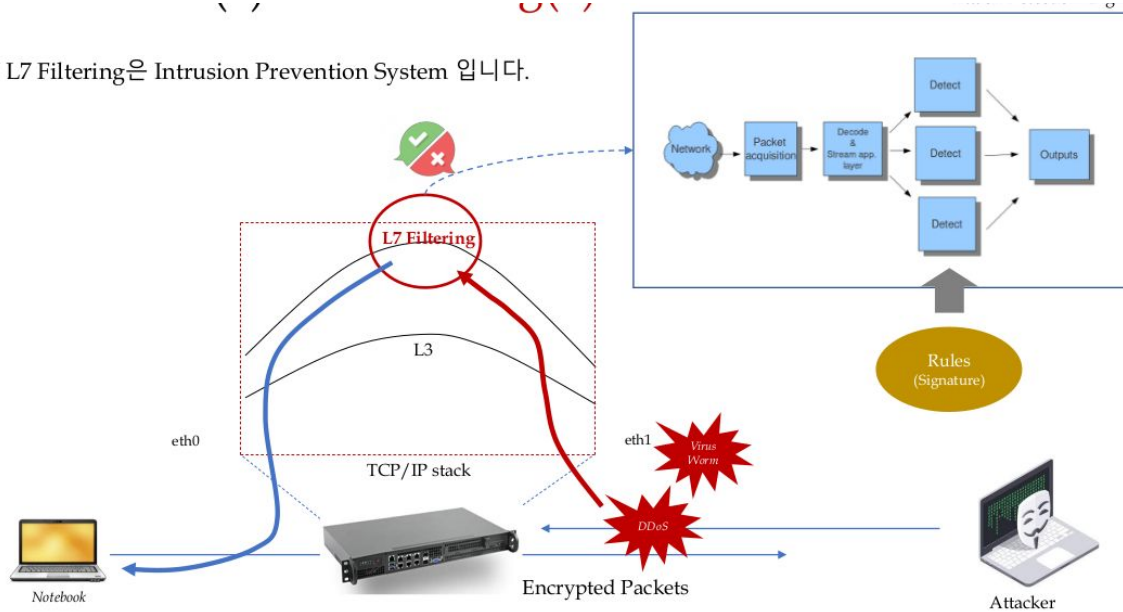
[그림 1.1] Suricata Home page

참고: IDS와 IPS의 차이는 inline 여부이다. 즉, inline mode(bridge or gateway 형태)를 사용하면 IPS(실시간 차단 개념)이고, 그렇지 않으면 IDS(실시간 detection 개념)이다.

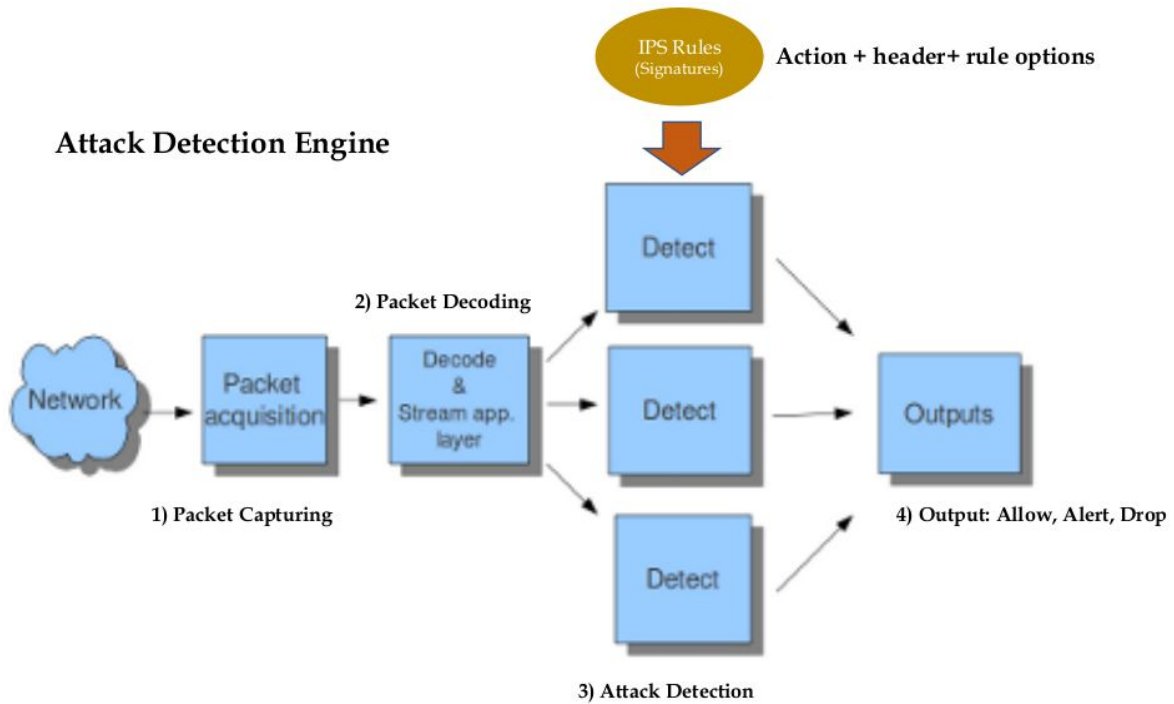


[그림 1.2] Suricata IPS 동작 원리 1

SPN L7 Filtering은 Intrusion Prevention System 입니다.



[그림 1.3] Suricata IPS 동작 원리 2



[그림 1.4] Suricata IPS 동작 원리 3



[그림 1.5] Suricata IPS 동작 원리 4

2. How to build and run Suricata

Suricata build와 관련해서는 아래 site를 참조할 수 있다.

<https://suricata.readthedocs.io/en/suricata-4.1.4/install.html>

Step-1) Ubuntu package 설치하기(Ubuntu 18.04 desktop 기준임)

```
$ sudo apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev \
    libnet1-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev \
    libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev \
    libnss3-dev libgeoip-dev liblua5.1-dev libhiredis-dev libevent-dev \
    python-yaml rustc cargo
```

```
$ sudo apt-get install libnetfilter-queue-dev libnetfilter-queue1 \
    libnetfilter-log-dev libnetfilter-log1 \
    libnfnetlink-dev libnfnetlink0
```

```
$ sudo apt install python3-distutils
```

```
$ sudo apt install liblzma-dev
```

```
$ sudo apt-get install liblz4-dev
```

Step-2) source code download & build 하기

<Latest code의 경우>

```
$ git clone https://github.com/OISF/suricata.git
```

```
$ cd suricata && git clone https://github.com/OISF/libhttp.git -b 0.5.x
```

```
$ ./autogen.sh
```

```
$ ./configure --prefix=/usr/ --sysconfdir=/etc/ --localstatedir=/var/ --enable-nfqueue
    → NFQUEUE와 함께 동작하도록 build
```

```
$ make clean && make
```

```
$ sudo make install-full
```

```
$ sudo ldconfig
```

<Stable code의 경우>

```
$ tar xzvf suricata-4.1.4.tar.gz
```

```
$ cd suricata-4.1.4
```

```
$ ./configure --prefix=/usr/ --sysconfdir=/etc/ --localstatedir=/var/ --enable-nfqueue
```

```
$ make
```

```
$ sudo make install
```

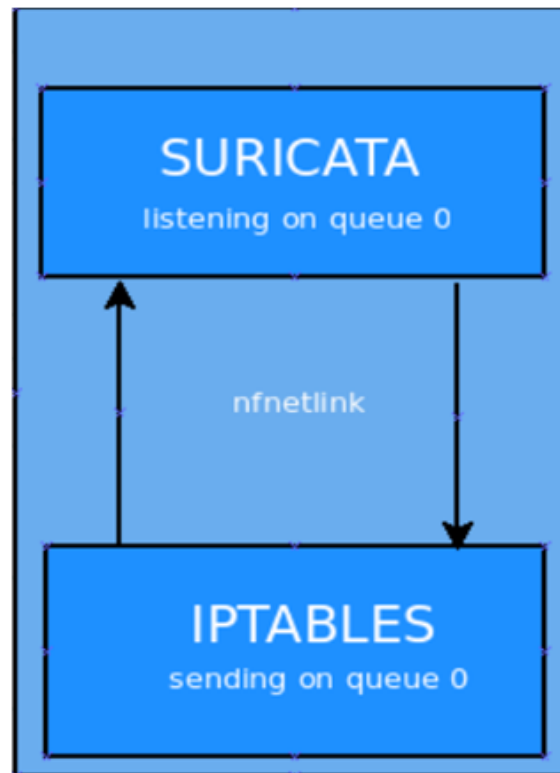
Step-3) Netfilter NFQUEUE rule 추가하기

모든 inbound/outbound/forward packet을 user space로 올려 주어야만, Suricata가 이를 받아 처리할 수가 있다. 이를 위한 iptables option으로는 -j NFQUEUE가 있다.

```
$ sudo iptables -I INPUT -m mark ! --mark 1/1 -j NFQUEUE --queue-num 0
```

```
$ sudo iptables -I OUTPUT -m mark ! --mark 1/1 -j NFQUEUE --queue-num 0
```

```
$ sudo iptables -I FORWARD -m mark ! --mark 1/1 -j NFQUEUE --queue-num 0
```



[그림 2.1] iptables와 Suricata의 상관 관계

Step-4) 실행하기

마지막으로 suricata configuration file을 수정한 후, 실행해 보도록 하자.

```
$ sudo vi /etc/suricata/suricata.yaml
```

```
...
```

```
nfq:
```

```
  mode: repeat # L7 filtering 후, NAT & SPN 처리 가능하게 하는 option임.
```

```
  repeat-mark: 1
```

```
  repeat-mark: 1
```

```
...
```

```
~
```

```
$ sudo suricata -c /etc/suricata/suricata.yaml -q 0 &
```

Step-5) 로그 확인하기


```
$ cd /var/log/suricata
$ ls -la
-rw-r--r-- 1 root root 175802 Jun 12 00:58 eve.json
-rw-r--r-- 1 root root    0 Jun 12 00:57 fast.log
-rw-r--r-- 1 root root 16461 Jun 12 00:58 stats.log
-rw-r--r-- 1 root root  4255 Jun 12 00:57 suricata.log
```

3. How to run Suricata on SPNBox

앞서 build한 suricata를 SPNBox에 올려 실행시켜 보도록 하자.

```
spnbox-c1037(config)# ips enable
spnbox-c1037(config)# sfirewall l7 fw insert any 0.0.0.0/0 any 0.0.0.0/0 any
spnbox-c1037(config)# sfirewall l7 in insert any 0.0.0.0/0 any 0.0.0.0/0 any
spnbox-c1037(config)# sfirewall l7 out insert any 0.0.0.0/0 any 0.0.0.0/0 any
```

```
spnbox-c1037(config)# show ips
IPS daemon(pid=2511) is alive.
```

```
=====
root    2511 38.0 3.4 943004 278612 ?    Ssl 00:57  0:04 /usr/bin/suricata -c
/etc/suricata/suricata
=====
```


4. Suricata IoT Cross Compile 하기

이번 절에서는 아래 site를 참조하여 Suricata-IoT 용 source code를 arm(aarch64)용으로 cross-compile해 보기로 하겠다.

<https://github.com/decanio/suricata-IoT/wiki/Suricata-IoT>

AARCH64용으로 compile하기

인터넷을 이곳 저곳 뒤져본 바, AARCH64용으로 compile하는 내용이 잘 정리된 부분을 찾기가 생각보다 어렵다.



```
$ export  
PATH=/home/chyi/workspace/spn/2ston_spnbox_prj/spnbox/boards/toolchain/gcc-li  
naro-5.2-2015.11-2-x86_64_aarch64-linux-gnu/bin:$PATH
```

<jansson-2.7>

<http://www.digip.org/jansson/releases/>

jansson-2.7.tar.bz2

```
$ ./configure --host=aarch64-linux-gnu  
--prefix=/home/chyi/workspace/downloads/Suricata/loT/output  
$ make  
$ make install
```

<libpcap-1.7.4>

<http://www.linuxfromscratch.org/blfs/view/7.9/basicnet/libpcap.html>

libpcap-1.7.4.tar.gz

```
$ ./configure --host=aarch64-linux-gnu --with-pcap=linux  
--prefix=/home/chyi/workspace/downloads/Suricata/loT/output  
$ make  
$ make install
```

<libyaml-0.1.6>

<https://pyyaml.org/wiki/LibYAML>

Yaml-0.2.2.tar.gz

```
$ ./configure --host=aarch64-linux-gnu  
--prefix=/home/chyi/workspace/downloads/Suricata/loT/output  
$ make  
$ make install
```

<libpcre - pcre3-8.38>

<https://ftp.pcre.org/pub/pcre/>

pcre-8.38.tar.bz2

```
$ ./configure --host=aarch64-linux-gnu  
--prefix=/home/chyi/workspace/downloads/Suricata/loT/output  
$ make  
$ make install
```

<libmagic - file-5.25>

<http://ftp.lfs-matrix.net/pub/clfs/conglomeration/file/>

file-5.25.tar.gz

```
$ ./configure --host=aarch64-linux-gnu  
--prefix=/home/chyi/workspace/downloads/Suricata/loT/output  
$ make  
→ 에러가 살짝 발생하는데, 무시해도 됨. 필요한 것은 이미 build 완료됨.  
$ make install
```

<suricata-loT>

```
$ git clone https://github.com/decanio/suricata-loT  
$ cd suricata-loT  
$ git checkout feature/loT-v7-riva
```

→ 이 상태에서 보니 IoT 관련 코드가 보이는데 ...

```
$ git clone https://github.com/OISF/libhttp
```

```
$ ./autogen.sh
```

```
$ ./configure --host=aarch64-linux-gnu \
--enable-nfqueue \
--with-libpcr-headers=\
    /home/chyi/workspace/downloads/Suricata/loT/output/include \
--with-libpcr-libraries=\
    /home/chyi/workspace/downloads/Suricata/loT/output/lib \
--with-libyaml-headers=\
    /home/chyi/workspace/downloads/Suricata/loT/output/include \
--with-libyaml-libraries=\
    /home/chyi/workspace/downloads/Suricata/loT/output/lib \
--with-libpcap-headers=\
    /home/chyi/workspace/downloads/Suricata/loT/output/include \
--with-libpcap-libraries=\
    /home/chyi/workspace/downloads/Suricata/loT/output/lib \
--with-libmagic-headers=\
    /home/chyi/workspace/downloads/Suricata/loT/output/include \
--with-libmagic-libraries=\
    /home/chyi/workspace/downloads/Suricata/loT/output/lib \
--with-libjansson-headers=\
    /home/chyi/workspace/downloads/Suricata/loT/output/include \
--with-libjansson-libraries=\
    /home/chyi/workspace/downloads/Suricata/loT/output/lib \
--with-libnetfilter_queue-headers=\
    /home/chyi/workspace/downloads/Suricata/loT/output/include \
--with-libnetfilter_queue-libraries=\
    /home/chyi/workspace/downloads/Suricata/loT/output/lib \
--with-libpcap-headers=\
    /home/chyi/workspace/downloads/Suricata/loT/output/include \
--with-libpcap-libraries=\
```

**/home/chyi/workspace/downloads/Suricata/loT/output/lib **
--prefix=/home/chyi/workspace/downloads/Suricata/loT/output

→ 아래와 같은 에러 발생하여 configure 파일 수정

ERROR! libnetfilter_queue library not found, go get it
from www.netfilter.org.
we automatically append libnetfilter_queue/ when searching
for headers etc. when the --with-libnfq-includes directive
is used

ERROR! libpcap library not found, go get it
from <http://www.tcpdump.org> or your distribution:

Ubuntu: apt-get install libpcap-dev
Fedora: yum install libpcap-devel

WARNING! libcap-ng library not found, go get it
from <http://people.redhat.com/sgrubb/libcap-ng/>
or your distribution:

Ubuntu: apt-get install libcap-ng-dev
Fedora: yum install libcap-ng-devel

Suricata will be built without support for dropping privs.

ERROR! libnspr library not found, go get it
from Mozilla or your distribution:

Ubuntu: apt-get install libnspr4-dev

Fedora: yum install nspr-devel

ERROR! libnss library not found, go get it
from Mozilla or your distribution:

Ubuntu: apt-get install libnss3-dev

Fedora: yum install nss-devel

ERROR! magic library not found, go get it
from <http://www.darwinsys.com/file/> or your distribution:

Ubuntu: apt-get install libmagic-dev

Fedora: yum install file-devel

checking for strlcat... no

checking zlib.h usability... no

checking zlib.h presence... no

checking for zlib.h... no

configure: error: zlib.h not found ...

configure: error: ./configure failed for libhttp

→ 결국 이 문제까지 오는데 ...

→ 아무리 봐도 **Suricata가 ARM cross compile** 가능하도록 신경을 많이 못 쓴 듯하다.

→ 일단, OpenWrt에서는 어찌되나 먼저 해 보자.

[참고 사항] GrapeBoard나 EspressoBin board의 경우는 직접 그 환경(ARM Ubuntu)에서 build하는게 답일 수도 있겠다.

OpenWrt용으로 compile하기

OpenWrt 용으로는 작업된 내용이 있는 듯 하다. 아래 site 내용을 참조해 보자.

<https://github.com/seanlinmt/suricata>

```
$ git clone https://github.com/seanlinmt/suricata
```

```
$ cp -r ./suricata  
/home/chyi/workspace/spn/2ston_spnbox_prj/spnbox/boards/glinet/openwrt/package/network/services
```

```
$ cd home/chyi/workspace/spn/2ston_spnbox_prj/spnbox/boards/glinet/openwrt/
```

```
$ make menuconfig
```

→ Network --> Firewall --> Suricata 선택

```
$ make -j1 V=99
```

→ 헐, 에러가 난다. 에러는 별도로 정리하지 않음.

→ 사실, Suricata를 Turris Omnia에서 돌리는 것은 여러가지 면에서 좀 비효율적이다.
괜한 시간 낭비하지 말고 이제 부터는 Snort를 검토해 보도록 하자.

5. Suricata Rules Update

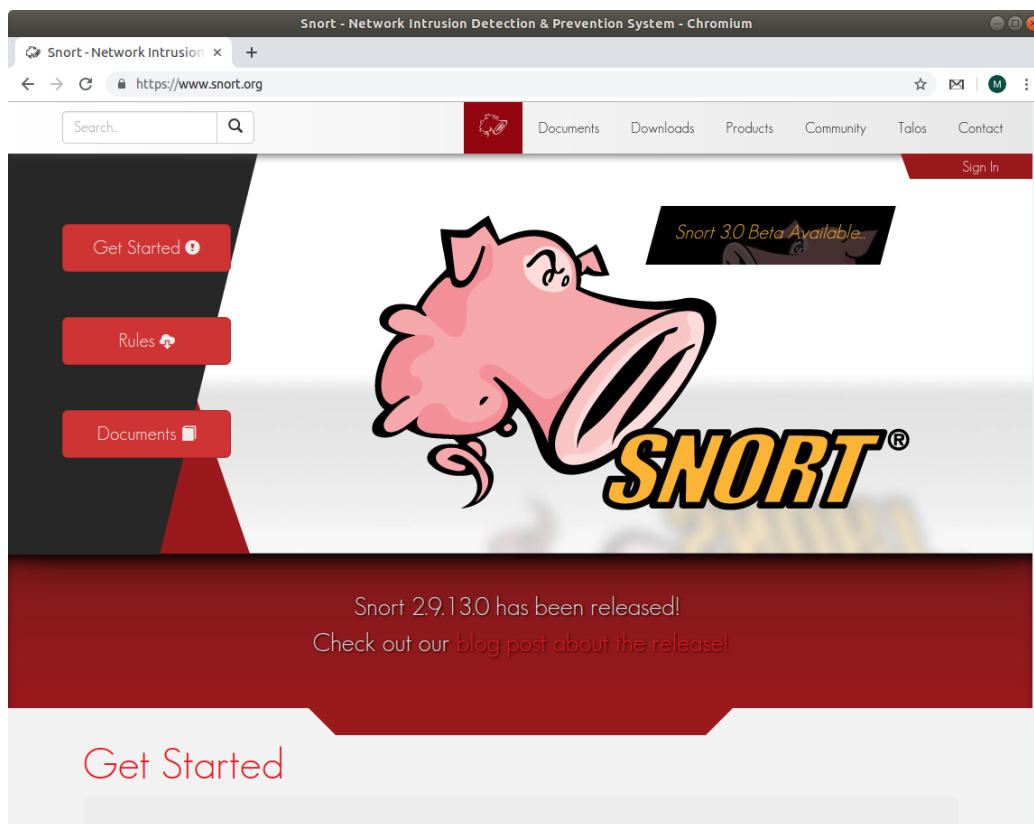
Suricata Rule은 계속 추가/갱신/삭제될 수 있다. 따라서 이를 주기적으로 관리하는 방안이 마련되어야 한다.

<TBD>

<https://www.howtoforge.com/tutorial/suricata-with-elk-and-web-front-ends-on-ubuntu-bionic-beaver-1804-lts/>

6. Snort Overview

아무래도 Suricata는 작은 시스템에 적합하지 않은 것 같다. 그렇다면 OpenWrt에도 이미 porting되어 있으며, ARM용으로 cross-compile하기도 비교적 수월한 SNORT를 검토해 보기로 하자.

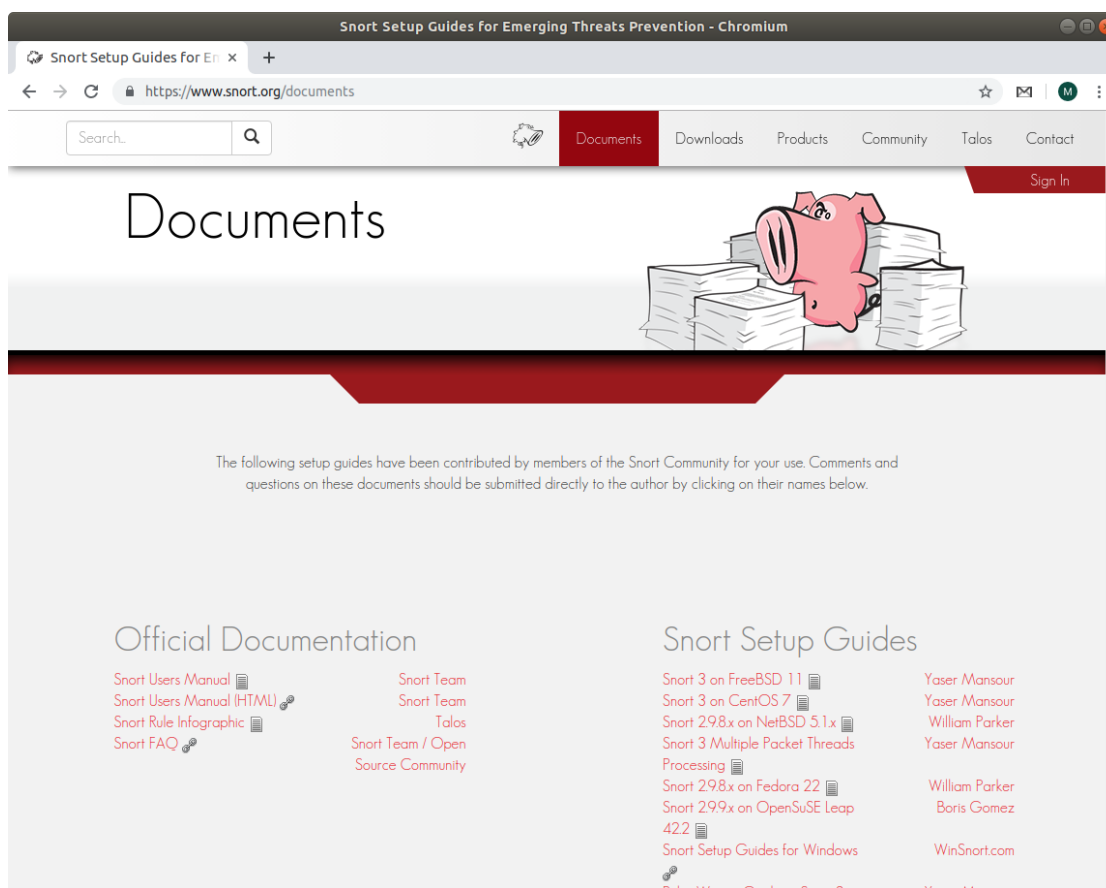


[그림 6.1] Snort home page

TABLE I. COMPARATIVE STUDY OF NIDS TOOL

Parameters	Snort	Suricata	Bro
Supported Platform	Win, MacOS, Unix	Win, MacOS, Unix	Unix like system, MacOS
License	GNU GPL V2	GNU GPL V2	BSD
IPS feature	Yes	Yes	No
PGP signed	Yes	Not Applicable	No
Support to high speed network	Medium	High	High
Configuration GUI	Yes	Yes	No
Offline Analysis	Yes for multiple files	Yes for single file	Yes for single file
Threads	Single Thread	Multithreaded	Single Thread
IPV6	Yes	Yes	No
Installation and Deployment	Easy	Easy	Difficult

위 표만 보아서는 high speed network 상황에서는 snort가 적합하지 않은 것 같다. 그렇다면 그 외의 상황에서는 쓸만하지 않을까 ?

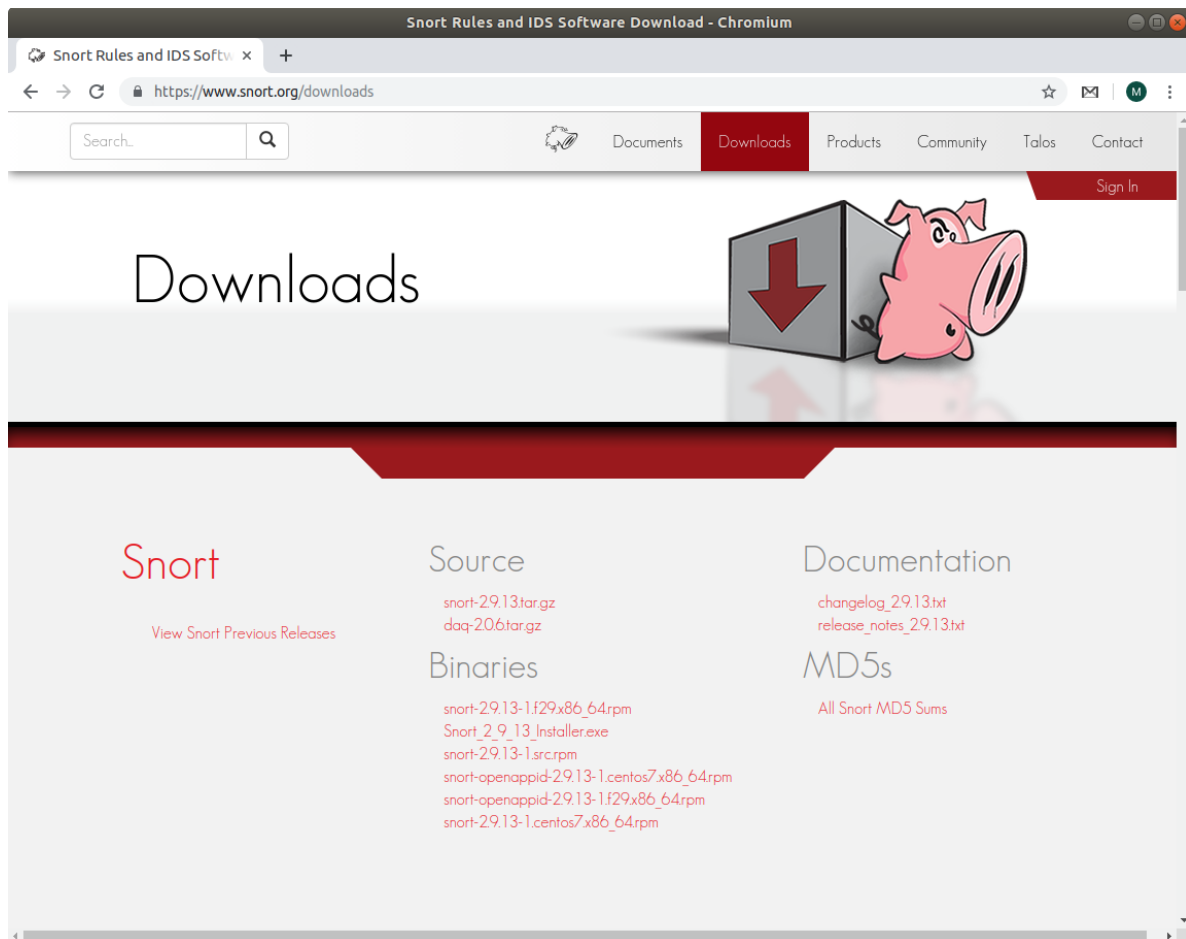


[그림 6.2] Snort home page - Documents 페이지

7. How to build & run Snort on SPNBox

아무래도 Suricata를 작은 system에 탑재하는데는 무리가 있어 보인다. 그렇다면 옛날로 돌아가서 Snort를 다시 검토해 보도록 하자. Snort가 Suricata 보다는 덜 복잡한 것 같다. 실제 binary 크기도 snort가 훨씬 더 작다. 또한, 참고적으로 OpenWrt에는 이미 Snort package가 포함되어 있다.

우선 아래 site에서 snort-2.9.13.tar.gz 파일과 daq-2.0.6.tar.gz 파일을 download 받도록 한다.

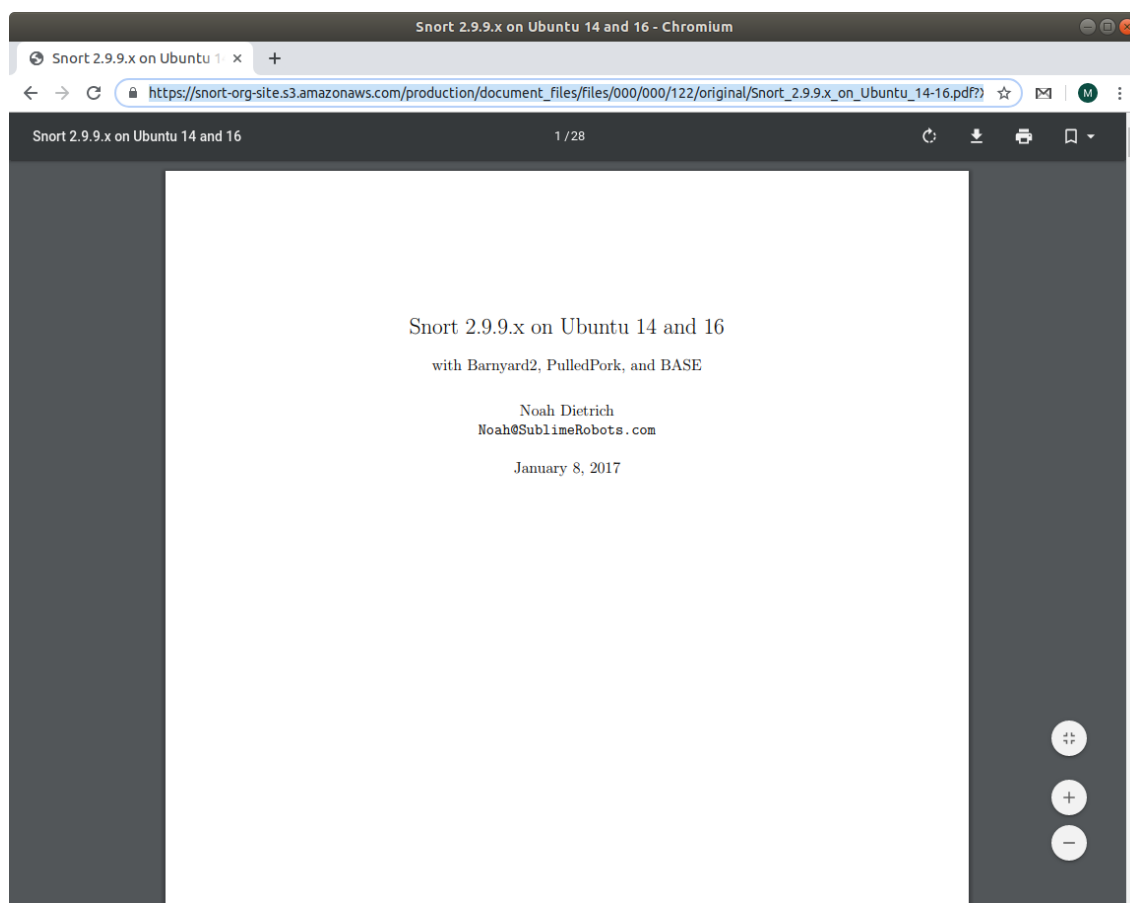


[그림 7.1] Snort download 공식 site

<Ubuntu 18.04에서 build해 보기>

먼저 snort build를 위해 몇가지 package를 내려 받도록 하자. 아래 내용은 [그림 7.2] 문서를 기초로 작성한 것이다.

https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/122/original/Snort_2.9.9.x_on_Ubuntu_14-16.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIAXACIED2SPMSC7GA%2F20190617%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20190617T024213Z&X-Amz-Expires=172800&X-Amz-SignedHeaders=host&X-Amz-Signature=d53cde8eb55a4f845d4c9b2529cd1cf9063f5006c2c7edce6640014f9170830c



[그림 7.2] Snort 2.9.9.x on Ubuntu 14 and 16

```
$ sudo apt-get install libpcap-dev libpcr3-dev libdumbnet-dev
$ sudo apt-get install libnetfilter-queue-dev
$ sudo apt-get install -y zlib1g-dev liblzma-dev openssl libssl-dev
  → 이미 설치되어 있는 상태 ...
$ sudo apt-get install bison flex
```

다음으로 daq library를 설치할 차례이다.

```
$ tar xvzf daq-2.0.6.tar.gz
$ cd daq-2.0.6
$ ./configure
```

...

```
Build AFPPacket DAQ module.. : yes
Build Dump DAQ module..... : yes
Build IPFW DAQ module..... : yes
Build IPQ DAQ module..... : no
Build NFQ DAQ module..... : yes //반드시 이 option이 켜져야 함.
Build PCAP DAQ module..... : yes
Build netmap DAQ module.... : no
```

```
$ make
```

```
$ sudo make install
```

이제 부터는 snort를 build해 보도록 하겠다.

```
$ tar xvzf snort-2.9.13.tar.gz
```

```
$ cd snort-2.9.13
```

```
$ ./configure
```

```
...
checking for luajit... no

ERROR! LuaJIT library not found. Go get it from http://www.luajit.org/ (or)
Try compiling without openAppId using '--disable-open-appid'
configure: error: "Fatal!"
```

→ configure 에러가 발생하였으니, 아래와 같이 option을 주기로 한다.

```
$ ./configure --disable-open-appid
```

```
$ make
```

```
$ sudo make install
```

```
$ sudo ldconfig
```

Snort 실행 준비하기

Snort build 및 파일 설치가 완료되었다. 근데, snort를 실제로 실행하기 위해서는 몇가지 설정 작업을 해야 한다. 왜 이렇게 불편하게 해 놓았는지 이해는 가지 않지만, 그래도 간단한 작업이므로, 아래 내용을 따라해 보기로 하자.

<snort 실행을 위한 디렉토리 생성 및 파일 복사 작업>

```
$ sudo mkdir /etc/snort
$ sudo mkdir /etc/snort/rules
$ sudo mkdir /etc/snort/rules/iplists
$ sudo mkdir /etc/snort/preproc_rules
$ sudo mkdir /usr/local/lib/snort_dynamicrules
$ sudo mkdir /etc/snort/so_rules

$ sudo touch /etc/snort/rules/iplists/black_list.rules
$ sudo touch /etc/snort/rules/iplists/white_list.rules
$ sudo touch /etc/snort/rules/local.rules
$ sudo touch /etc/snort/sid-msg.map

$ sudo mkdir /var/log/snort
$ sudo mkdir /var/log/snort/archived_logs

$ sudo chmod -R 5775 /etc/snort
$ sudo chmod -R 5775 /var/log/snort
$ sudo chmod -R 5775 /var/log/snort/archived_logs
$ sudo chmod -R 5775 /etc/snort/so_rules
$ sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules

$ cd etc
root@spnbox2:~/workspace/SNORT/snort-2.9.13/etc$ sudo cp *.conf* /etc/snort/
root@spnbox2:~/workspace/SNORT/snort-2.9.13/etc$ sudo cp *.map /etc/snort/
```

```

root@spnbox2:~/workspace/SNORT/snort-2.9.13/etc$ sudo cp *.dtd /etc/snort/

$ cd src/dynamic-preprocessors/build/usr/local/lib/snort_dynamicpreprocessor
$ sudo cp * /usr/local/lib/snort_dynamicpreprocessor/

$ sudo sed -i "s/include \$RULE_PATH/#include \$RULE_PATH/"
/etc/snort/snort.conf
    → 현재 include rule을 모두 막아 버림.
    → PulledPork를 사용하기 위함 ==> rule을 하나의 파일로 통합.

$ sudo vi /etc/snort/snort.conf
    → see michael string

```

<Snort 실행에 필요한 디렉토 정리>

→ 아래 내용이 모두 준비되었다면, 이제 부터 snort를 실행할 수가 있겠다.

Snort binary file: /usr/local/bin/snort

Snort configuration file: /etc/snort/snort.conf

Snort log data directory: /var/log/snort

Snort rules directories: /etc/snort/rules

/etc/snort/so rules

/etc/snort/preproc rules

/usr/local/lib/snort dynamicrules

Snort IP list directories: /etc/snort/rules/iplists

Snort dynamic preprocessors: /usr/local/lib/snort dynamicpreprocessor/

tree 명령으로 snort 관련 준비 사항을 체크해 보아도 좋겠다.

\$ tree /etc/snort

```

/etc/snort
├── attribute_table.dtd

```

```

├── classification.config
├── file_magic.conf
├── gen-msg.map
├── preproc_rules
├── reference.config
├── rules
│   ├── iplist
│   │   ├── black_list.rules
│   │   └── white_list.rules
│   └── local.rules
├── sid-msg.map
├── snort.conf
├── so_rules
├── threshold.conf
└── unicode.map

```

지금까지 수작업으로 설정한 내용이 정상적인지를 snort 명령을 통해 확인해 보도록 하자.

```
spnbox@spnbox-c1037:/etc/snort$ sudo snort -T -i br0 -c /etc/snort/snort.conf
```

Running in Test mode

==== Initializing Snort ===

Initializing Output Plugins!

Initializing Preprocessors!

Initializing Plug-ins!

Parsing Rules file "/etc/snort/snort.conf"

PortVar 'HTTP_PORTS' defined : [80:81 311 383 591 593 901 1220 1414 1741 1830 2301
2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779

8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800
8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555
]

PortVar 'SHELLCODE_PORTS' defined : [0:79 81:65535]

PortVar 'ORACLE_PORTS' defined : [1024:65535]

PortVar 'SSH_PORTS' defined : [22]

PortVar 'FTP_PORTS' defined : [21 2100 3535]

PortVar 'SIP_PORTS' defined : [5060:5061 5600]

PortVar 'FILE_DATA_PORTS' defined : [80:81 110 143 311 383 591 593 901 1220 1414
1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145
7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243
8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444
41080 50002 55555]

PortVar 'GTP_PORTS' defined : [2123 2152 3386]

Detection:

Search-Method = AC-Full-Q

Split Any/Any group = enabled

Search-Method-Optimizations = enabled

Maximum pattern length = 20

Tagged Packet Limit: 256

Loading dynamic engine /usr/local/lib/snort_dynamicengine/libsf_engine.so... done

Loading all dynamic detection libs from /usr/local/lib/snort_dynamicrules...

WARNING: No dynamic libraries found in directory /usr/local/lib/snort_dynamicrules.

Finished Loading all dynamic detection libs from /usr/local/lib/snort_dynamicrules

Loading all dynamic preprocessor libs from /usr/local/lib/snort_dynamicpreprocessor/...

Loading dynamic preprocessor library


/usr/local/lib/snort_dynamicpreprocessor//libsf_ssh_preproc.so... done

Loading dynamic preprocessor library

/usr/local/lib/snort_dynamicpreprocessor//libsf_smtp_preproc.so... done

Loading dynamic preprocessor library

/usr/local/lib/snort_dynamicpreprocessor//libsf_ssl_preproc.so... done



```

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_reputation_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_dce2_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_gtp_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_dnp3_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_dns_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_sdf_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_pop_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_modbus_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_ftptelnet_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_sip_preproc.so... done

Loading dynamic preprocessor library
/usr/local/lib/snort_dynamicpreprocessor//libsf_imap_preproc.so... done

Finished Loading all dynamic preprocessor libs from
/usr/local/lib/snort_dynamicpreprocessor/

Log directory = /var/log/snort

WARNING: ip4 normalizations disabled because not inline.
WARNING: tcp normalizations disabled because not inline.
WARNING: icmp4 normalizations disabled because not inline.
WARNING: ip6 normalizations disabled because not inline.
WARNING: icmp6 normalizations disabled because not inline.

Frag3 global config:
    Max frags: 65536

```

Fragment memory cap: 4194304 bytes

Frag3 engine config:

Bound Address: default

Target-based policy: WINDOWS

Fragment timeout: 180 seconds

Fragment min_ttl: 1

Fragment Anomalies: Alert

Overlap Limit: 10

Min fragment Length: 100

Max Expected Streams: 768

Stream global config:

Track TCP sessions: ACTIVE

Max TCP sessions: 262144

TCP cache pruning timeout: 30 seconds

TCP cache nominal timeout: 180 seconds

Memcap (for reassembly packet storage): 8388608

Track UDP sessions: ACTIVE

Max UDP sessions: 131072

UDP cache pruning timeout: 30 seconds

UDP cache nominal timeout: 180 seconds

Track ICMP sessions: INACTIVE

Track IP sessions: INACTIVE

Log info if session memory consumption exceeds 1048576

Send up to 2 active responses

Wait at least 5 seconds between responses

Protocol Aware Flushing: ACTIVE

Maximum Flush Point: 16000

Stream TCP Policy config:

Bound Address: default

Reassembly Policy: WINDOWS

Timeout: 180 seconds

Limit on TCP Overlaps: 10

Maximum number of bytes to queue per session: 1048576

Maximum number of segs to queue per session: 2621

Options:

Require 3-Way Handshake: YES

3-Way Handshake Timeout: 180

Detect Anomalies: YES

Reassembly Ports:

21 client (Footprint)

22 client (Footprint)

23 client (Footprint)

25 client (Footprint)

42 client (Footprint)

53 client (Footprint)

79 client (Footprint)

80 client (Footprint) server (Footprint)

81 client (Footprint) server (Footprint)

109 client (Footprint)

110 client (Footprint)

111 client (Footprint)

113 client (Footprint)

119 client (Footprint)

135 client (Footprint)

136 client (Footprint)

137 client (Footprint)

139 client (Footprint)

143 client (Footprint)

161 client (Footprint)

additional ports configured but not printed.

Stream UDP Policy config:

Timeout: 180 seconds

HttpInspect Config:

GLOBAL CONFIG

Detect Proxy Usage: NO

IIS Unicode Map Filename: /etc/snort/unicode.map

IIS Unicode Map Codepage: 1252

Memcap used for logging URI and Hostname: 150994944

Max Gzip Memory: 838860

Max Gzip Sessions: 1613

Gzip Compress Depth: 65535

Gzip Decompress Depth: 65535

DEFAULT SERVER CONFIG:

Server profile: All

Ports (PAF): 80 81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037
3128 3702 4343 4848 5250 6988 7000 7001 7144 7145 7510 7777 7779 8000 8008 8014
8028 8080 8085 8088 8090 8118 8123 8180 8181 8243 8280 8300 8800 8888 8899 9000
9060 9080 9090 9091 9443 9999 11371 34443 34444 41080 50002 55555

Server Flow Depth: 0

Client Flow Depth: 0

Max Chunk Length: 500000

Small Chunk Length Evasion: chunk size <= 10, threshold >= 5 times

Max Header Field Length: 750

Max Number Header Fields: 100

Max Number of WhiteSpaces allowed with header folding: 200

Inspect Pipeline Requests: YES

URI Discovery Strict Mode: NO

Allow Proxy Usage: NO

Disable Alerting: NO

Oversize Dir Length: 500

Only inspect URI: NO

Normalize HTTP Headers: NO

Inspect HTTP Cookies: YES

Inspect HTTP Responses: YES

Extract Gzip from responses: YES

Decompress response files:

Unlimited decompression of gzip data from responses: YES

Normalize Javascripts in HTTP Responses: YES

Max Number of WhiteSpaces allowed with Javascript Obfuscation in HTTP responses:
200

Normalize HTTP Cookies: NO

Enable XFF and True Client IP: NO

Log HTTP URI data: NO

Log HTTP Hostname data: NO

Extended ASCII code support in URI: NO

Ascii: YES alert: NO

Double Decoding: YES alert: NO

%U Encoding: YES alert: YES

Bare Byte: YES alert: NO

UTF 8: YES alert: NO

IIS Unicode: YES alert: NO

Multiple Slash: YES alert: NO

IIS Backslash: YES alert: NO

Directory Traversal: YES alert: NO

Web Root Traversal: YES alert: NO

Apache WhiteSpace: YES alert: NO

IIS Delimiter: YES alert: NO

IIS Unicode Map: GLOBAL IIS UNICODE MAP CONFIG

Non-RFC Compliant Characters: 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07

Whitespace Characters: 0x09 0x0b 0x0c 0x0d

Legacy mode: NO

rpc_decode arguments:

Ports to decode RPC on: 111 32770 32771 32772 32773 32774 32775 32776 32777
32778 32779

alert_fragments: INACTIVE

alert_large_fragments: INACTIVE

alert_incomplete: INACTIVE

alert_multiple_requests: INACTIVE

FTPTelnet Config:

GLOBAL CONFIG

Inspection Type: stateful

Check for Encrypted Traffic: YES alert: NO

Continue to check encrypted data: YES

TELNET CONFIG:

Ports: 23

Are You There Threshold: 20

Normalize: YES

Detect Anomalies: YES

FTP CONFIG:

FTP Server: default

Ports (PAF): 21 2100 3535

Check for Telnet Cmds: YES alert: YES

Ignore Telnet Cmd Operations: YES alert: YES

Ignore open data channels: NO

FTP Client: default

Check for Bounce Attacks: YES alert: YES

Check for Telnet Cmds: YES alert: YES

Ignore Telnet Cmd Operations: YES alert: YES

Max Response Length: 256

SMTP Config:

Ports: 25 465 587 691

Inspection Type: Stateful

Normalize: ATRN AUTH BDAT DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN EVFY
EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND STARTTLS
SOML TICK TIME TURN TURNME VERB VRFY X-EXPS XADR XAUTH XCIR XEXCH50 XGEN
XLICENSE X-LINK2STATE XQUE XSTA XTRN XUSR CHUNKING X-ADAT X-DRCP X-ERCP
X-EXCH50

Ignore Data: No

Ignore TLS Data: No

Ignore SMTP Alerts: No

Max Command Line Length: 512

Max auth Command Line Length: 1000

Max Specific Command Line Length:

ATRN:255 AUTH:246 BDAT:255 DATA:246 DEBUG:255

EHLO:500 EMAL:255 ESAM:255 ESND:255 ESOM:255

ETRN:246 EVFY:255 EXPN:255 HELO:500 HELP:500

IDENT:255 MAIL:260 NOOP:255 ONEX:246 QUEU:246

QUIT:246 RCPT:300 RSET:246 SAML:246 SEND:246

SIZE:255 STARTTLS:246 SOML:246 TICK:246 TIME:246

TURN:246 TURNME:246 VERB:246 VRFY:255 X-EXPS:246

XADR:246 XAUTH:246 XCIR:246 XEXCH50:246 XGEN:246


XLICENSE:246 X-LINK2STATE:246 XQUE:246 XSTA:246 XTRN:246

XUSR:246

Max Header Line Length: 1000

Max Response Line Length: 512

X-Link2State Alert: Yes



Drop on X-Link2State Alert: No
Alert on commands: None
Alert on unknown commands: No
SMTP Memcap: 838860
MIME Max Mem: 838860
Base64 Decoding: Enabled
Base64 Decoding Depth: Unlimited
Quoted-Printable Decoding: Enabled
Quoted-Printable Decoding Depth: Unlimited
Unix-to-Unix Decoding: Enabled
Unix-to-Unix Decoding Depth: Unlimited
Non-Encoded MIME attachment Extraction: Enabled
Non-Encoded MIME attachment Extraction Depth: Unlimited
Log Attachment filename: Enabled
Log MAIL FROM Address: Enabled
Log RCPT TO Addresses: Enabled
Log Email Headers: Enabled
Email Hdrs Log Depth: 1464
SSH config:
Autodetection: ENABLED
Challenge-Response Overflow Alert: ENABLED
SSH1 CRC32 Alert: ENABLED
Server Version String Overflow Alert: ENABLED
Protocol Mismatch Alert: ENABLED
Bad Message Direction Alert: DISABLED
Bad Payload Size Alert: DISABLED
Unrecognized Version Alert: DISABLED
Max Encrypted Packets: 20
Max Server Version String Length: 100

MaxClientBytes: 19600 (Default)

Ports:

22

DCE/RPC 2 Preprocessor Configuration

Global Configuration

DCE/RPC Defragmentation: Enabled

Memcap: 102400 KB

Events: co

SMB Fingerprint policy: Disabled

Server Default Configuration

Policy: WinXP

Detect ports (PAF)

SMB: 139 445

TCP: 135

UDP: 135

RPC over HTTP server: 593

RPC over HTTP proxy: None

Autodetect ports (PAF)

SMB: None

TCP: 1025-65535

UDP: 1025-65535

RPC over HTTP server: 1025-65535

RPC over HTTP proxy: None

Invalid SMB shares: C\$ D\$ ADMIN\$

Maximum SMB command chaining: 3 commands

SMB file inspection: Disabled

DNS config:

DNS Client rdata txt Overflow Alert: ACTIVE

Obsolete DNS RR Types Alert: INACTIVE

Experimental DNS RR Types Alert: INACTIVE

Ports: 53

SSLPP config:

Encrypted packets: not inspected

Ports:

443	465	563	636	989
992	993	994	995	7801
7802	7900	7901	7902	7903
7904	7905	7906	7907	7908
7909	7910	7911	7912	7913
7914	7915	7916	7917	7918
7919	7920			

Server side data is trusted

Maximum SSL Heartbeat length: 0

Sensitive Data preprocessor config:

Global Alert Threshold: 25

Masked Output: DISABLED

SIP config:

Max number of sessions: 40000

Max number of dialogs in a session: 4 (Default)

Status: ENABLED

Ignore media channel: DISABLED

Max URI length: 512

Max Call ID length: 80

Max Request name length: 20 (Default)

Max From length: 256 (Default)

Max To length: 256 (Default)

Max Via length: 1024 (Default)

Max Contact length: 512

Max Content length: 2048

Ports:

5060 5061 5600

Methods:

invite cancel ack bye register options refer subscribe update join info message
notify benotify do qauth sprack publish service unsubscribe prack

IMAP Config:

Ports: 143

IMAP Memcap: 838860

MIME Max Mem: 838860

Base64 Decoding: Enabled

Base64 Decoding Depth: Unlimited

Quoted-Printable Decoding: Enabled

Quoted-Printable Decoding Depth: Unlimited

Unix-to-Unix Decoding: Enabled

Unix-to-Unix Decoding Depth: Unlimited

Non-Encoded MIME attachment Extraction: Enabled

Non-Encoded MIME attachment Extraction Depth: Unlimited

POP Config:

Ports: 110

POP Memcap: 838860

MIME Max Mem: 838860

Base64 Decoding: Enabled

Base64 Decoding Depth: Unlimited

Quoted-Printable Decoding: Enabled

Quoted-Printable Decoding Depth: Unlimited

Unix-to-Unix Decoding: Enabled

Unix-to-Unix Decoding Depth: Unlimited

Non-Encoded MIME attachment Extraction: Enabled

Non-Encoded MIME attachment Extraction Depth: Unlimited

Modbus config:

Ports:

502

DNP3 config:

Memcap: 262144

Check Link-Layer CRCs: ENABLED

Ports:

20000

Reputation config:

WARNING: Can't find any whitelist/blacklist entries. Reputation Preprocessor disabled.

+++++

Initializing rule chains...

0 Snort rules read

0 detection rules

0 decoder rules

0 preprocessor rules

0 Option Chains linked into 0 Chain Headers

+++++

+-----[Rule Port Counts]-----

	tcp	udp	icmp	ip
src	0	0	0	0
dst	0	0	0	0
any	0	0	0	0
nc	0	0	0	0
s+d	0	0	0	0

+-----

```

+-----[detection-filter-config]-----
| memory-cap : 1048576 bytes
+-----[detection-filter-rules]-----
| none

```

```

+-----[rate-filter-config]-----
| memory-cap : 1048576 bytes
+-----[rate-filter-rules]-----
| none

```

```

+-----[event-filter-config]-----
| memory-cap : 1048576 bytes
+-----[event-filter-global]-----
+-----[event-filter-local]-----
| none
+-----[suppression]-----
| none

```

Rule application order: pass->drop->sdrop->reject->alert->log

Verifying Preprocessor Configurations!

pcap DAQ configured to passive.

Acquiring network traffic from "br0".

--== Initialization Complete ==--

„_ -*> Snort! <*-

o")~ Version 2.9.13 GRE (Build 15013)

"" 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 1.8.1

Using PCRE version: 8.39 2016-06-14

Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 3.1 <Build 1>

Preprocessor Object: SF_IMAP Version 1.0 <Build 1>

Preprocessor Object: SF_SIP Version 1.1 <Build 1>

Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>

Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>

Preprocessor Object: SF_POP Version 1.0 <Build 1>

Preprocessor Object: SF_SDF Version 1.1 <Build 1>

Preprocessor Object: SF_DNS Version 1.1 <Build 4>

Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>

Preprocessor Object: SF_GTP Version 1.1 <Build 1>

Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>

Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>

Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>

Preprocessor Object: SF_SMTP Version 1.1 <Build 9>

Preprocessor Object: SF_SSH Version 1.1 <Build 3>

Snort successfully validated the configuration!

Snort exiting

Snort 실행하기

정말로, 모든 것이 준비되었다. 그렇다면, 이제 부터 Snort를 실행해 보도록 하자. Suricata의 경우와 마찬가지로 NFQ가 enable 되어 있는 상태에서 snort를 실행해 주어야만 원하는 결과를 얻을 수 있다. 단, 순서는 snort를 먼저 띄우고, iptables 설정을 해 주어야만 된다.

<NFQ inline mode로 snort 구동하기>

```
$ sudo snort -Q --daq nfq --daq-var device=br0 --daq-var queue=1 -c /etc/snort/snort.conf
```

→ OK, 동작한다. 자세한 로그는 생략 함.

```
$ sudo iptables -I FORWARD -j NFQUEUE --queue-num 1
```

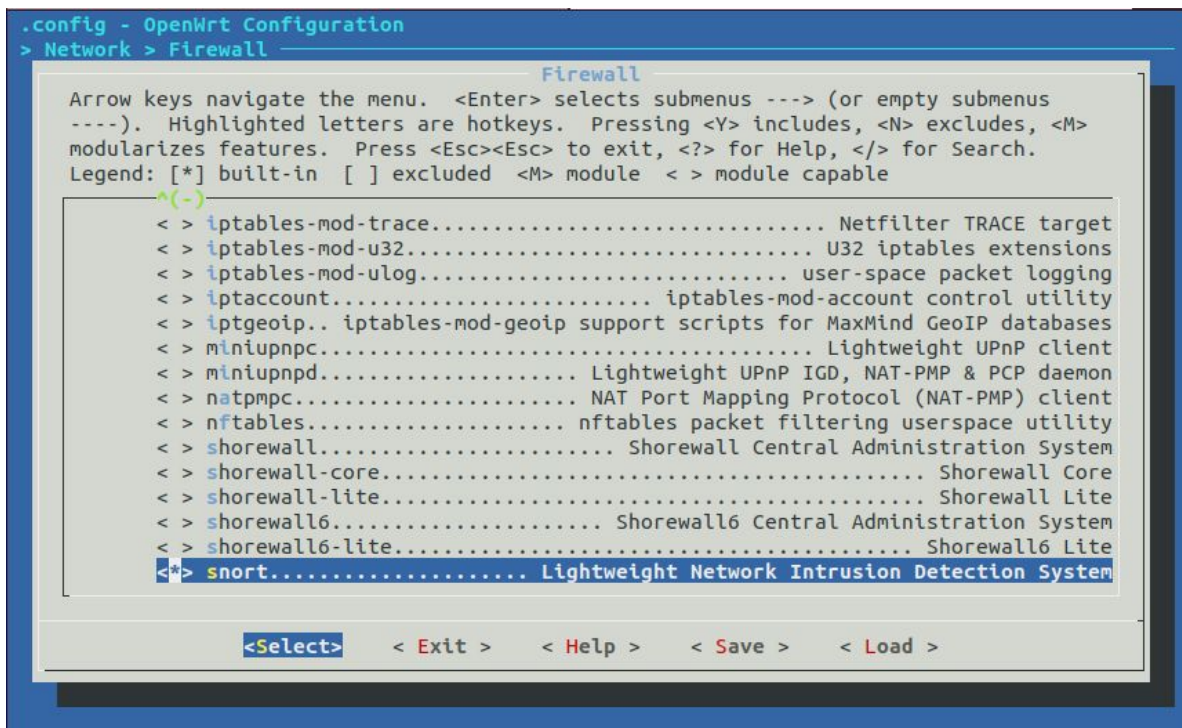
[참고 사항] 지금까지 설명한 내용을 SPNBox에 탑재할 것인가, 말 것인가? 고민이 된다. 어차피 X86 series에는 Suricata가 탑재될 것이고, OpenWrt를 사용하는 Access Point 제품에서는 아래 8장에서 설명하겠지만, 이미 snort가 porting되어 있으니 말이다...

8. How to build Snort on OpenWrt

```
$ cd home/chyi/workspace/spn/2ston_spnbox_prj/spnbox/boards/glinet/openwrt/
```

```
$ make menuconfig
```

→ Network --> Firewall --> Snort 선택



[그림 8.1] OpenWrt menuconfig - snort 선택

\$ make -j1 V=99

➔ 정상적으로 build가 된다.

<결과 확인>

\$ find . -name "snort" -print

```
./build_dir/target-mips_24kc_musl/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/.pkgdir/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/.pkgdir/snort/usr/bin/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/.pkgdir/snort/etc/config/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/.pkgdir/snort/etc/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/.pkgdir/snort/etc/init.d/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/src/dynamic-preprocessors/build/
usr/lib/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/src/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-mips_24kc/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-mips_24kc/snort/usr/bi
n/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-mips_24kc/snort/etc/config/s
nort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-mips_24kc/snort/etc/snort
```

```

./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-mips_24kc/snort/etc/init.d/s
nort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-install/usr/share/doc/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-install/usr/lib/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-install/usr/bin/snort
./build_dir/target-mips_24kc_musl/snort/snort-2.9.11.1/ipkg-install/usr/include/snort
./build_dir/target-mips_24kc_musl/root.orig-ar71xx/usr/bin/snort
./build_dir/target-mips_24kc_musl/root.orig-ar71xx/etc/config/snort
./build_dir/target-mips_24kc_musl/root.orig-ar71xx/etc/snort
./build_dir/target-mips_24kc_musl/root.orig-ar71xx/etc/init.d/snort
./build_dir/target-mips_24kc_musl/root-ar71xx/usr/bin/snort
./build_dir/target-mips_24kc_musl/root-ar71xx/etc/config/snort
./build_dir/target-mips_24kc_musl/root-ar71xx/etc/snort
./build_dir/target-mips_24kc_musl/root-ar71xx/etc/init.d/snort
./staging_dir/target-mips_24kc_musl/usr/lib/snort
./staging_dir/target-mips_24kc_musl/usr/include/snort
./staging_dir/target-mips_24kc_musl/root-ar71xx/usr/bin/snort
./staging_dir/target-mips_24kc_musl/root-ar71xx/etc/config/snort
./staging_dir/target-mips_24kc_musl/root-ar71xx/etc/snort
./staging_dir/target-mips_24kc_musl/root-ar71xx/etc/init.d/snort
./package/feeds/packages/snort
./feeds/packages/net/snort

```

이후 Gl.iNet board에서 snort를 실행하는 것과 관련해서는 별도로 정리하지 않았다. 이 부분은 나중에 정말 필요하면 따로 테스트해 볼 예정이다.

9. Snort Rule Update

Snort의 경우는 기본 source에 rule 파일을 포함시키지 않고, PulledPork라는 것을 사용하여 1개의 rule 형태로 관리하고 있는 듯 보인다. 이와 관련하여 정리해 보고자 한다.

<TBD>

