

네트워크 시그니처 개발

- 부제 : Snort Rule 개발의 정석
- 일시 : 2023.11.24
- 강사 : 정보보호전문가



Snort Rule

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"SERVER-WEBAPP mojoPortal Forums txtTitle cross site scripting attempt";  
flow:to_server,established; content:"/Forums/EditForum.aspx"; fast_pattern:only; http_uri; content:"txtTitle="; http_client_body;  
pcre:"/(%24|%(25)?24)txtTitle=[^&]*?([%22%27%3c%3e%28%29]|%(25)?(22|27|3c|3e|28|29)|script|onload|src)/Pi"; metadata:policy max-  
detect-ips drop, policy security-ips drop, service http; reference:url,www.exploit-db.com/exploits/49184; classtype:attempted-user; sid:61082;  
rev:1;)
```

오늘의 목표

- 1 Snort Rule을 개발할 수 있는 환경 만들기
- 2 Snort Rule 구조 이해하기
- 3 정보보안기사 Snort 문제 풀기
- 4 다른 사람이 개발한 Snort Rule 해석하기
- 5 Snort Rule 개발하기



Snort 소개

스노트(Snort)는 마틴 로시(Martin Roesch)가 1998년에 개발한 오픈 소스 네트워크 침입 탐지 시스템(**IDS**: Intrusion Detection System)이자, 침입 차단 시스템(**IPS**: Intrusion Prevention System)으로 Sourcefire에서 개발하였습니다.

현재는 시스코가 Sourcefire를 2013년에 인수하면서 Snort를 소유하고 있으며, 2021년 1월 19일 Snort 3을 공식 출시 후 꾸준히 업데이트를 하고 있습니다.



IDS vs IPS

침입 탐지 시스템 (IDS)

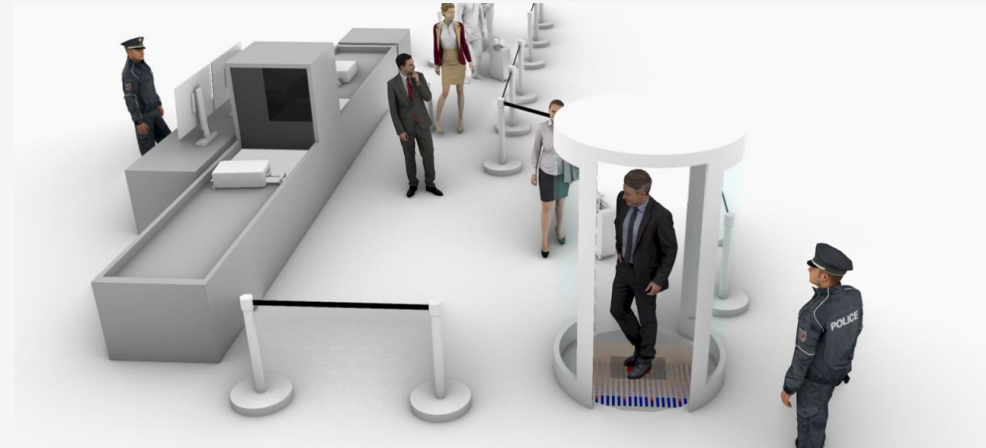


1. 방화벽을 통과하는 위협 트래픽 분석
2. 패킷 미러링(Mirroring) 기술
3. 방화벽과 연동하여 공격자 IP 차단

* 용어 설명

- 패킷 미러링 : 패킷을 복사하는 기술

침입 차단 시스템 (IPS)



1. One-Way 공격에 대응하기 위해 개발
2. 패킷 인라인(in-line) 기술
3. 제로데이 취약점을 대응하는데 효과적

* 용어 설명

- One-Way 공격 : 패킷 1개로 영향을 주는 공격
- 제로데이 취약점 : 취약점 패치가 되지 않는 공격

네트워크 탐지 시그니처

네트워크 탐지 시그니처(Signature)는 침입 탐지 시스템(IDS), 침입 차단 시스템(IPS)과 같이 네트워크 트래픽을 모니터링하는 솔루션에서 네트워크 패킷을 분석하여 해킹 공격을 탐지하기 위해 등록하는 규칙을 말합니다.

도로 교통법으로 비유해 보면, 도로에서는 **차량의 종류, 속도, 방향, 신호** 등을 근거로 **교통 위반 여부를 판단**합니다.

마찬가지로 SNORT에서는 **패킷에서 시그니처(특정 위협의 고유한 문자열 또는 행동)**을 기반으로 **공격 여부를 판단**합니다. 즉, 시그니처는 도로 교통법규의 세부 사항과 같이 공격의 특징을 정의하는 역할을 합니다.



개발 환경 구축

1 윈도우용 패킷 디코딩 도구인 **Wireshark**를 설치해 주세요.

wireshark.org 접속 > **Download Wireshark Now** 클릭 > **Windows x64 Installer** 다운로드 > 기본 설치 진행

- URL : <https://1.as.dl.wireshark.org/win64/Wireshark-4.2.0-x64.exe>

2 윈도우용 침입 탐지 시스템인 **Snort**를 설치해 주세요.

snort.org 접속 > **Downloads** 페이지로 이동 > (Snort 2) **Snort_2_9_20_Installer.x64.exe** 다운로드 > 기본 설치 진행

- URL : https://snort.org/downloads/snort/Snort_2_9_20_Installer.x64.exe

개발 환경 구축 : Snort 환경 설정 (1/3)

1 윈도우 환경에 맞게 **snort.conf** 파일 수정해 주세요.

C:\Snort\etc > 에디터 프로그램(예:[Notepad++](#))으로 **snort.conf** 파일을 열어서 내용 수정하기

241 라인

```
#####  
# Step #4: Configure dynamic loaded libraries.  
# For more information, see Snort Manual, Configuring Snort - Dynamic Modules  
#####  
  
# path to dynamic preprocessor libraries  
dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor  
  
# path to base preprocessor engine  
dynamicengine C:\Snort\lib\snort_dynamicengine/sf_engine.dll  
  
# path to dynamic rules libraries  
dynamicdetection directory C:\Snort\preproc_rules
```


개발 환경 구축 : Snort 환경 설정 (2/3)

1 윈도우 환경에 맞게 **snort.conf** 파일 수정해 주세요.

local.rules 파일을 제외하고 모든 Snort rules 파일을 주석 처리하고, C:\Snort\rules 경로에 local.rules 파일을 생성해 주세요.

* Tip : 문자열 바꾸기 기능을 이용하면 편해요. (include \$RULE_PATH → # include \$RULE_PATH)

538 라인

```
#####  
# Step #7: Customize your rule set  
# For more information, see Snort Manual, Writing Snort Rules  
#  
# NOTE: All categories are enabled in this conf file  
#####  
  
# site specific rules  
include $RULE_PATH/local.rules  
  
# include $RULE_PATH/app-detect.rules  
# include $RULE_PATH/attack-responses.rules
```

개발 환경 구축 : Snort 환경 설정 (3/3)

2 윈도우 환경에 맞게 필요한 파일을 생성해 주세요.

C:\Snort\rules 경로에 **local.rules** 파일을 생성해 주세요. (빈 파일)

C:\Snort\rules 경로에 **white_list.rules** 파일을 생성해 주세요. (빈 파일)

C:\Snort\rules 경로에 **black_list.rules** 파일을 생성해 주세요. (빈 파일)

3 윈도우 Snort가 잘 동작하는지 테스트해 주세요.

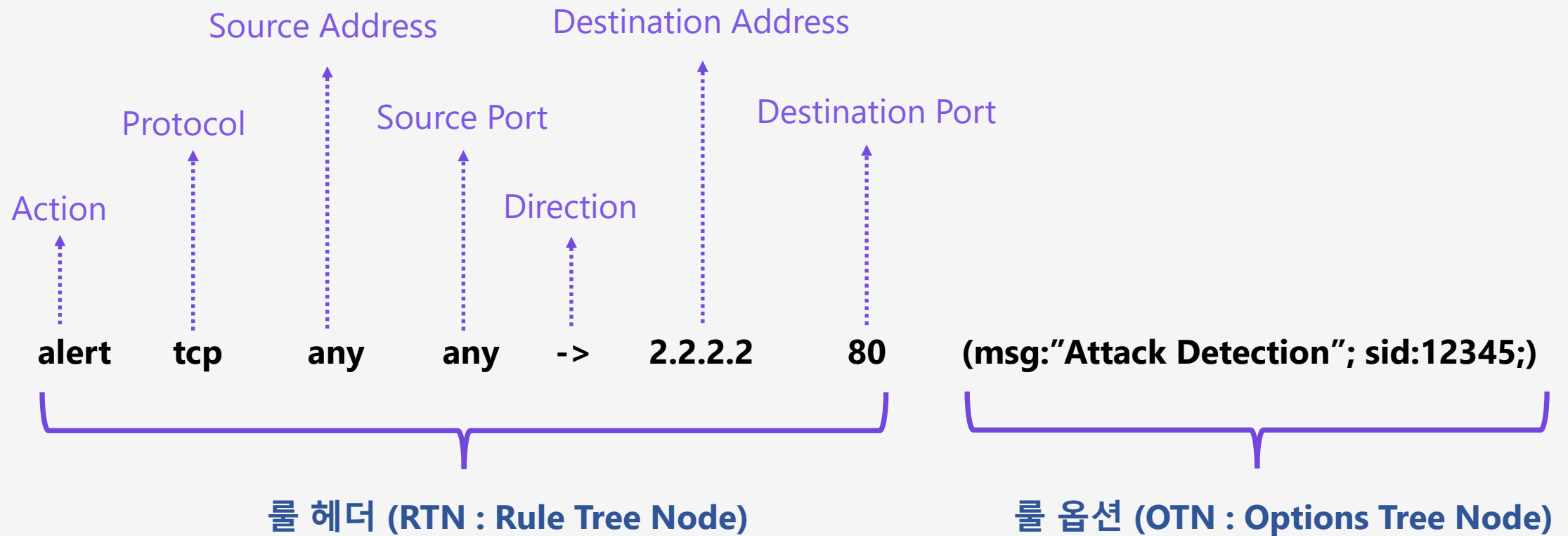
C:\Snort 경로에 **pcap** 경로를 생성해 주세요.

네트워크 패킷 파일을 다운로드 받아서 C:\Snort\pcap 경로에 넣어주세요.
(<https://github.com/packetinside/pcap/blob/main/sample.zip>) *passwd: pcap

CMD 창을 열고, C:\Snort 경로로 이동하여 아래 명령어를 실행해 보세요.

```
C:\Snort>bin\snort.exe -q -A console -c etc\snort.conf -r pcap\sample.pcap
```

Snort Rule 구조



룰 헤더

SNORT 룰 헤더(RTN: Rule Tree Node)에는 7가지 옵션이 있습니다.

- ① Action
- ② Protocol
- ③ Source IP
- ④ Source Port
- ⑤ Direction
- ⑥ Destination IP
- ⑦ Destination Port

룰 헤더는 ACL(Access Control List) 방화벽과 유사하게 패킷 헤더의 정보를 기반으로 네트워크 패킷을 필터링할 수 있습니다.

그래서 잘 알려진 공격자 IP 목록(일명 블랙리스트)를 적용하여 탐지하는데, 활용할 수도 있습니다.

룰 헤더 : Action

SNORT가 시그니처 조건에 맞는 패킷을 탐지했을 때, 어떻게 처리할 지 결정하는데 사용합니다.

Action	경고 발생	로그 저장	패킷 차단	비고
☆ alert	○	○		
log		○		
pass				
drop		○	○	
reject	○		○	<ul style="list-style-type: none">TCP는 reset 전송UDP는 ICMP port unreachable 메시지 전송
sdrop			○	

alert **tcp** **any** **any** **->** **2.2.2.2** **80** **(msg:"Attack Detection"; sid:12345;)**

룰 헤더 : Protocol

Protocol 항목은 검사할 프로토콜 종류를 선택하는데 사용합니다.

- ① tcp : 전송 제어 프로토콜(*Transmission Control Protocol, TCP*)
- ② ucp : 사용자 데이터그램 프로토콜(*User Datagram Protocol, UDP*)
- ③ icmp : 인터넷 제어 메시지 프로토콜(*Internet Control Message Protocol, ICMP*)
- ④ ip : 인터넷 프로토콜(*Internet Protocol, IP*)

alert tcp any any -> 2.2.2.2 80 (msg:"Attack Detection"; sid:12345;)

룰 헤더 : Source/Destination Address

IP Address 항목은 출발지 IP와 목적지 IP를 지정하는데 사용합니다.

IP 주소 형태(예: 2.2.2.2)뿐만 아니라 변수명 형태(예: \$HOME_NET)로도 지정할 수 있습니다.

SNORT 환경 설정 파일(snort.conf)에서 내부 IP 대역을 나타내는 'HOME_NET'과 외부 IP 대역을 나타내는 'EXTERNAL_NET' 등 여러 개의 변수명이 이미 정의되어 있어서 내 네트워크 환경에 맞게 수정해 주시면 됩니다.

44 라인

```
# Setup the network addresses you are protecting
#ipvar HOME_NET any
ipvar HOME_NET 2.2.2.2/24          <----- 내부 IP 대역으로 수정

# Set up the external network addresses. Leave as "any" in most situations
#ipvar EXTERNAL_NET any
ipvar EXTERNAL_NET !$HOME_NET     <----- 내부 IP 대역 이외는 외부 IP로 설정
```

```
alert tcp $EXTERNAL_NET any -> 2.2.2.2 80 (msg:"Attack Detection"; sid:12345;)
```

룰 헤더 : Source/Destination Port

Port 항목은 출발지 Port와 목적지 Port를 지정하는데 사용합니다.

Port 형태는 숫자뿐만 아니라 변수명 형태(예: \$HTTP_PORTS)로도 지정할 수 있습니다.

SNORT 환경 설정 파일(snort.conf)에는 HTTP 포트를 나타내는 'HOME_PORTS' 외에도 여러가지의 변수명이 사전에 정의되어 있습니다.

74 라인

```
# List of ports you run web servers on
portvar HTTP_PORTS
[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,
7777,7779,8000,8008,8014,8028,8080,8085,8088,8090,8118,8123,8180,8181,8243,8280,8300,8800,8888,8899,9000,9060,9080,9090,9
091,9443,9999,11371,34443,34444,41080,50002,55555]

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80
```

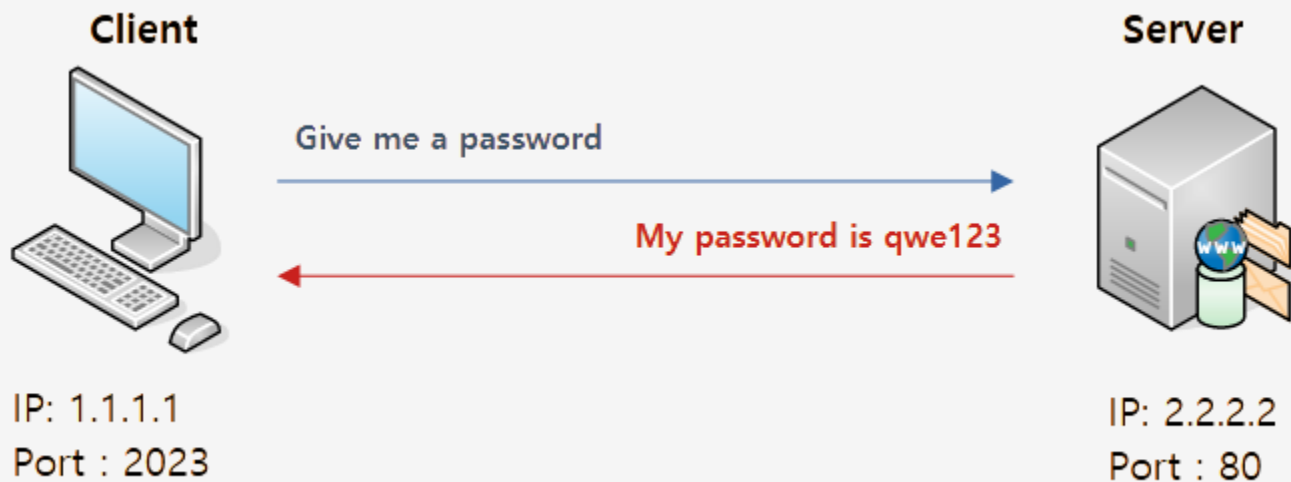
alert tcp any any -> 2.2.2.2 \$HTTP_PORTS (msg:"Attack Detection"; sid:12345;)

룰 헤더 : Direction

Direction 항목은 네트워크 양방향 통신에서 어느 방향의 패킷을 검사할지 지정할 수 있습니다.

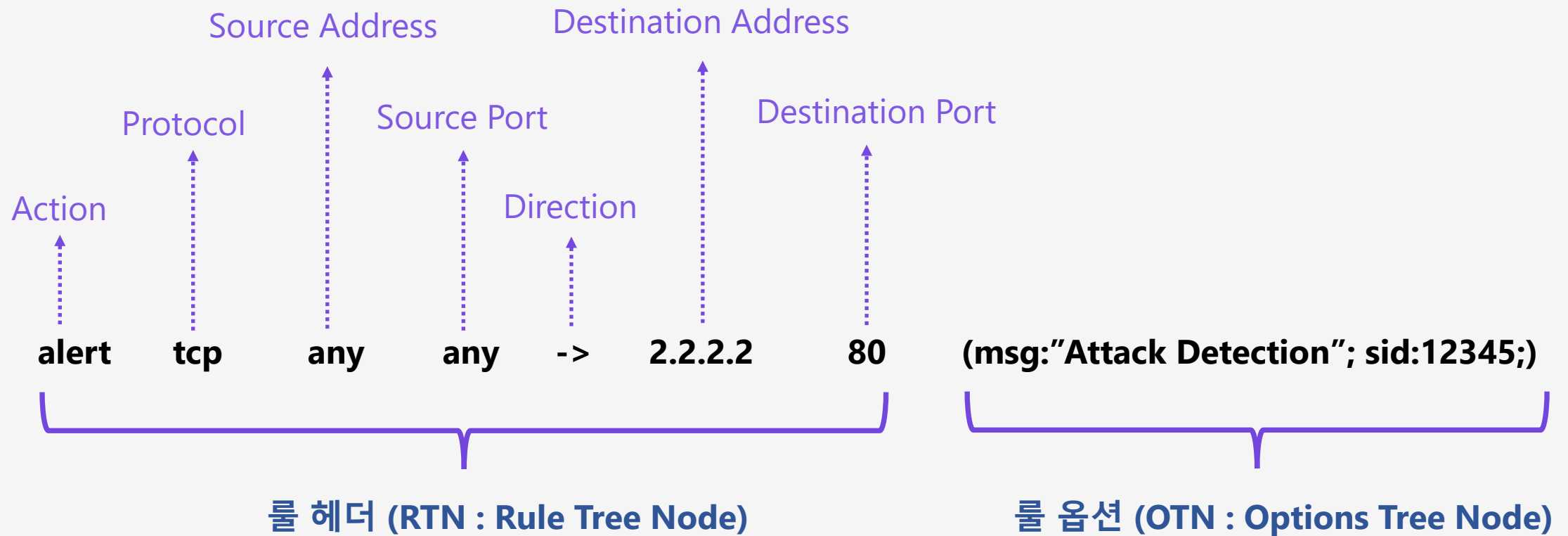
단방향 연산자(->)는 선택한 방향의 패킷만 검사합니다.

반면에, 양방향 연산자(<>)를 사용하면 양쪽 방향을 모두 검사할 수 있습니다.



```
alert tcp any any -> 2.2.2.2 80 (msg:"Attack Detection"; sid:12345;)
```

Snort Rule 구조



룰 옵션

SNORT 룰 옵션(OTN: Option Tree Node)는 패킷의 헤더와 페이로드에서 패턴 매칭 할 수 있는 여러가지 키워드를 제공합니다.

주의 깊게 봐야 할 부분은 다음과 같습니다.

- 패킷 헤더의 패턴 매칭 키워드
 - Wireshark를 통해 검사할 수 있는 영역 확인
- 패킷 페이로드의 패턴 매칭 키워드
 - 문자열의 매칭 포인트의 이동 변화

룰 옵션의 시작과 끝은 소괄호로 표시하고, 키워드 구분자는 세미콜론(;)을 사용합니다.

탐지 문자열을 입력할 수 있는 키워드(content, uricontent, pcre)에는 쌍따옴표(")로 묶어줘야 합니다.

```
alert tcp any any -> any 80 (msg:"Event Name"; content:"Hello"; pcre:"/010-\d{3}/"; sid:12345;)
```

Part I. 룰 관련 정보 키워드

룰 옵션 : msg

msg 키워드는 패턴 매칭이 되었을 때 보여 줄 메시지를 설정하는데 사용합니다.

즉, 탐지 이벤트명을 작성하는 키워드라고 생각하시면 됩니다.

- ① msg:"Apache Log4j logging Remote Code Execution Attempt";
(해석) 아파치 프로그램의 Log4 관련 취약점이며, 원격에서 코드 실행이 가능한 공격 탐지
- ② msg:"Trojan.Agent outbound connection attempt";
(해석) 트로이잔 악성코드로 외부로 접속을 시도하는 행위를 탐지
- ③ msg:"Ransomware.CryptoLocker variant download attempt"
(해석) 랜섬웨어 크립토락커 변종 악성코드가 다운로드 되는 행위를 탐지

룰 옵션 : metadata

metadata 키워드는 자유롭게 추가 정보를 입력하는데 사용합니다.

입력 정보는 키-값 형식으로 입력하고 키와 값은 공백으로 구분하고, 여러 개의 키-값을 입력하는 경우에는 쉼표(,)로 구분합니다.

- ① metadata:cvss 9.0, att&ck TA0002, cyber-kill-chain 3, service http;
- ② metadata:policy max-detect-ips drop, policy security-ips drop, service http;

룰 옵션 : reference

reference 키워드는 시그니처에 대한 정보를 입력하는데 사용합니다.

취약점을 탐지하는 시그니처인 경우 CVE 정보와 취약점에 대해 설명이 있는 URL 주소를 넣고, 악성코드 관련 시그니처는 바이러스토탈(www.virustotal.com)에서 다수의 안티바이러스(백신) 엔진으로 분석한 결과를 URL 주소로 넣는 경우가 많습니다.

String	URL Prefix
☆ url	http://
☆ cve	http://cve.mitre.org/cgi-bin/cvename.cgi?name=
nessus	http://cgi.nessus.org/plugins/dump.php3?id=
arachnids	http://www.whitehats.com/info/IDS
mcafee	http://vil.nai.com/vil/content/v_
osvdb	http://osvdb.org/show/osvdb/
msb	http://technet.microsoft.com/en-us/security/bulletin/
bugtraq	http://www.securityfocus.com/bid/

① reference:url,www.exploit-db.com/exploits/51747; reference:cve,2023-32707;

룰 옵션 : classtype

classtype 키워드는 시그니처의 공격 유형을 정의하는데 사용합니다. 공격의 유형 정보는 classification.conf 설정 파일에 정의되어 있으며, 등록된 값 중 shortname을 사용하면 됩니다.

```
# config classification:shortname,short description,priority  
  
config classification: not-suspicious,Not Suspicious Traffic,3  
config classification: unknown,Unknown Traffic,3  
config classification: bad-unknown,Potentially Bad Traffic, 2  
config classification: attempted-recon,Attempted Information Leak,2  
config classification: successful-recon-limited,Information Leak,2
```

- ① classtype:attempted-recon;
- ② classtype:attempted-user;

룰 옵션 : sid

sid 키워드는 SNORT에서 시그니처를 식별하는데 사용합니다.

중복되거나 사용범위를 넘지 않는 고유한 숫자로 입력해야 합니다.

sid의 사용범위는 아래와 같습니다.

범위	설명
100 미만	시스템에 예약된 범위
100~999999	Snort에서 배포하는 시그니처 범위
1000000 이상	사용자가 사용할 수 있는 범위

① sid:12345;

룰 옵션 : rev

rev 키워드는 시그니처를 수정한 횟수를 기록하는데 사용합니다.

일반적으로 오탐이 발생하거나 미탐이 발생하여 기존 시그니처를 개선할 때, 시그니처를 수정하게 되는데, 이 때 rev(revision) 값을 1씩 증가시켜 줍니다.

그래서 rev 정보를 보면 몇 번이나 수정되었는지 알 수 있습니다.

참고로, rev 키워드의 값은 1부터 시작합니다.

① rev:5;

룰 옵션 : flow (1/2)

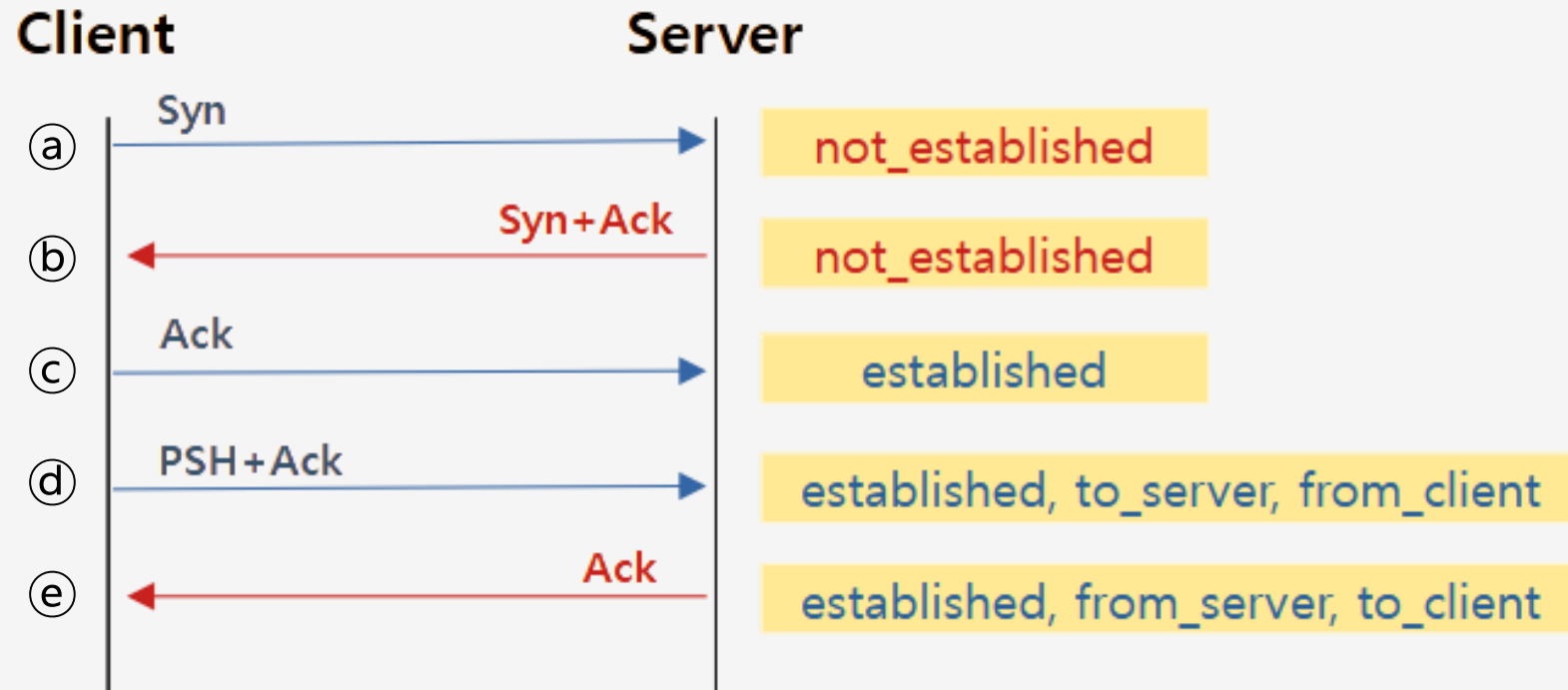
flow 키워드는 TCP 세션의 연결 상태와 패킷의 방향 등을 설정하는데 사용합니다.

참고로, 주로 사용하는 옵션에 대해서는 ☆로 표시하였습니다.

구분	옵션	옵션 설명
TCP 세션 연결 상태	☆ established	TCP 세션이 연결된 상태의 패킷 검사
	not_established	TCP 세션이 연결되지 않은 상태의 패킷 검사
	stateless	TCP 세션 상태와 상관없이 패킷 검사
패킷 방향	☆ to_server	서버로 전송되는 패킷 검사
	☆ from_server	서버가 응답하는 패킷 검사
	☆ to_client	클라이언트로 전송되는 패킷 검사 (from_server와 동일)
	☆ from_client	클라이언트에서 요청하는 패킷 검사 (to_server와 동일)
스트림 상태	no_stream	스트림(세그먼트 된 TCP 패킷 조합)이 아닌 패킷 검사
	only_stream	스트림(세그먼트 된 TCP 패킷 조합) 상태의 패킷 검사
단편화 상태	no_frag	단편화(조각난 패킷)되지 않은 패킷 검사
	only_frag	단편화(조각난 패킷)된 패킷 검사

룰 옵션 : flow (2/2)

TCP 통신과 flow 옵션의 상태 변화



① flow:to_server,established;

② flow:from_server,established

Part II. 프로토콜 헤더 검사하기

룰 옵션 : IP 헤더 구조

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7								
Version				Header Length				Type of Service								Total Length																							
Identification																Flags				Fragment Offset																			
Time to Live								Protocol								Header Checksum																							
Source Address																																							
Destination Address																																							

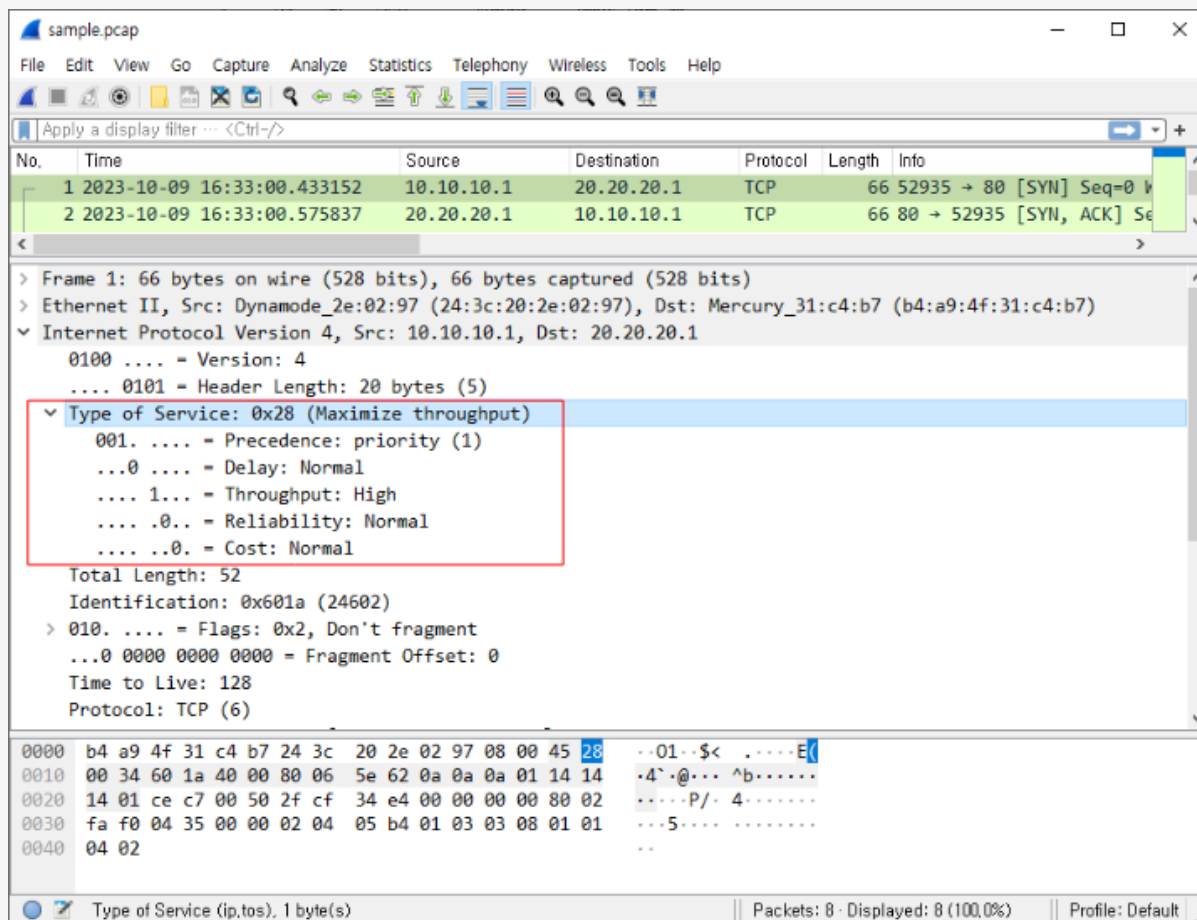
룰 옵션 : tos

tos 키워드는 IPv4 헤더의 Type of Service(TOS) 필드 값을 검사하는데 사용합니다.

TOS 필드는 네트워크에서 패킷의 우선순위를 관리하는데 중요한 역할을 합니다. 예를 들어, 어떤 패킷이 높은 우선순위를 가지고 있다면, 라우터는 이 패킷을 더 빠르게 처리하거나 전달할 수 있습니다.

■ 입력 형식

tos:[!]<number>;



```
alert ip any any -> any any (msg:"IP Header tos keyword"; tos:40; sid:1000001;)
```

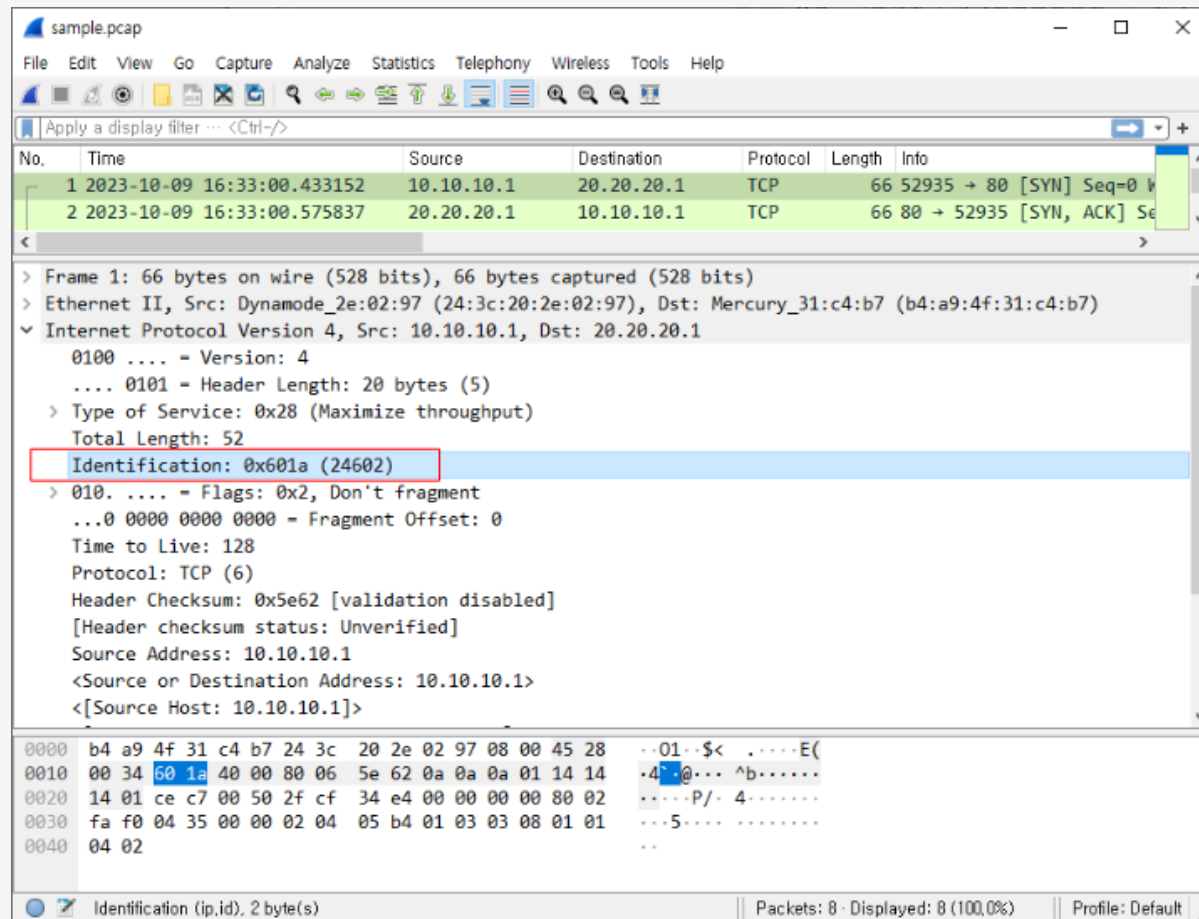
룰 옵션 : id

id 키워드는 IPv4 헤더의 Identification(ID) 필드 값을 검사하는데 사용합니다.

ID 필드는 패킷을 구별하기 위해 사용합니다. 송신자가 패킷을 분할/단편화(Fragmentation)하게 되면, 이들 패킷에는 동일한 ID 번호가 부여됩니다. 수신자는 동일한 ID 번호를 가진 패킷들을 확인하고, 이를 바탕으로 원래의 패킷으로 재조합(Reassembly)을 할 수 있게 됩니다.

■ 입력 형식

id:<number>;



```
alert ip any any -> any any (msg:"IP Header id keyword"; id:24602; sid:1000002;)
```

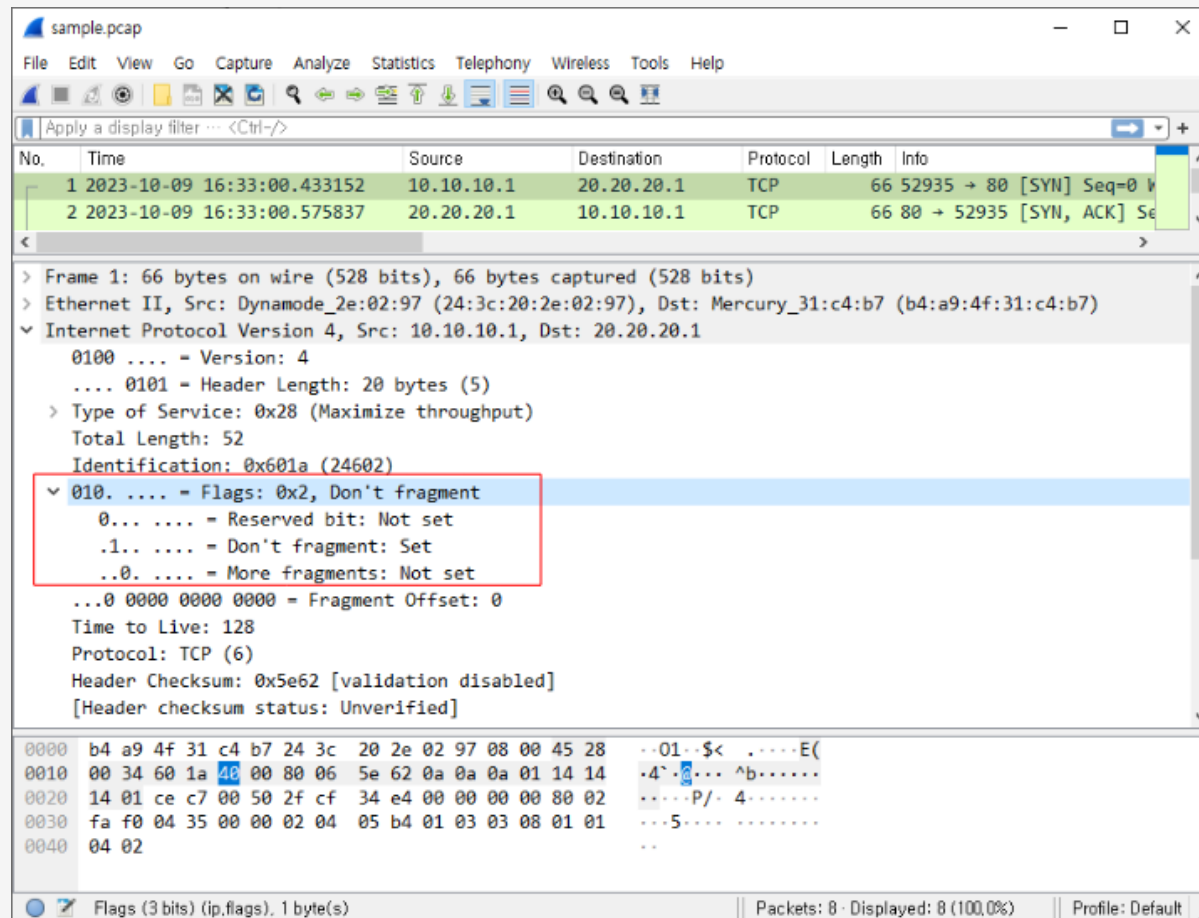

룰 옵션 : fragbits

fragbits 키워드는 IP 헤더의 Frags 값을 확인하는데 사용합니다.

Flags 필드는 패킷의 분할/단편화(Fragmentation) 여부와 추가 패킷이 있다는 것을 수신자에게 알려줘서 패킷 재조합(Reassembly)을 할 수 있습니다.

■ 입력 형식

fragbits:[+*!]<[RDM]>;



alert ip any any -> any any (msg:"IP Header fragbits keyword"; fragbits:M; sid:1000003;)

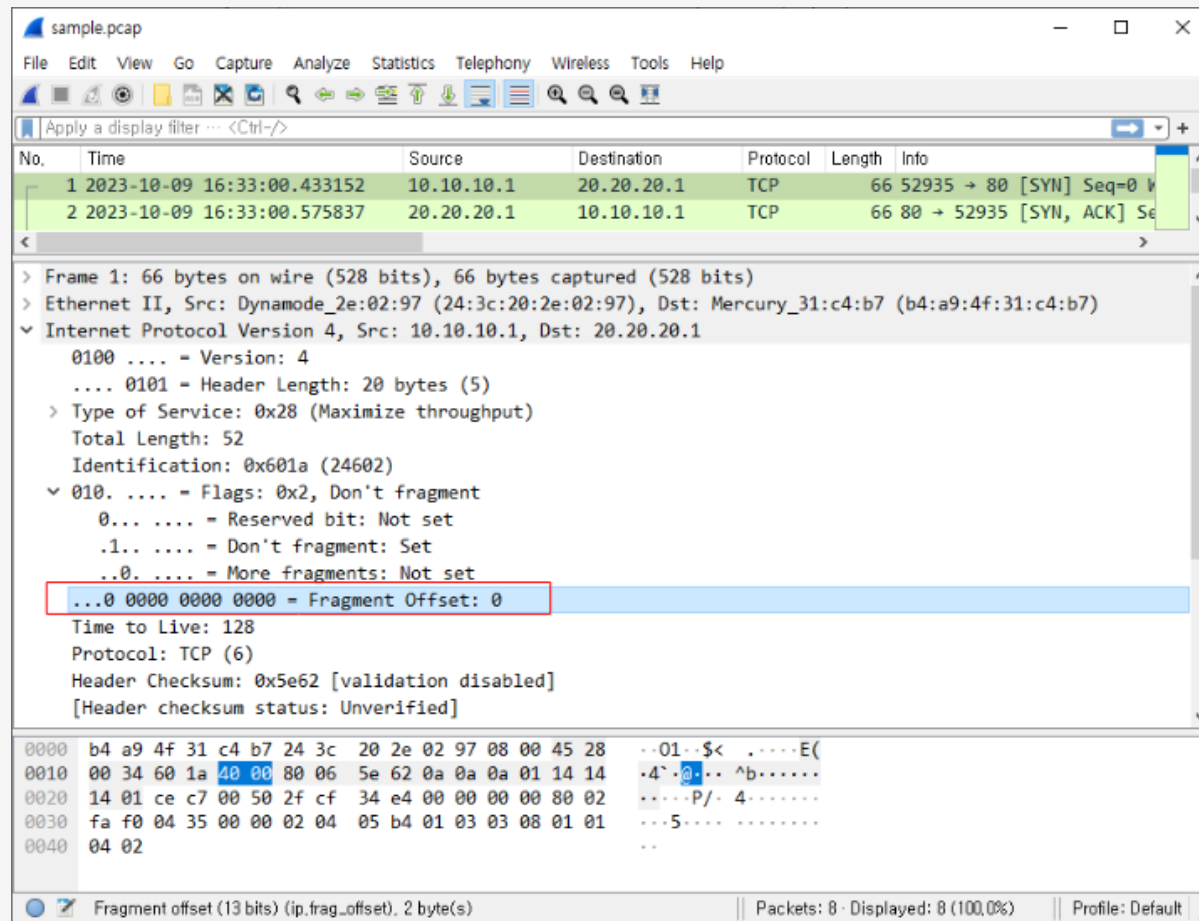
룰 옵션 : fragoffset

fragoffset 키워드는 IP 헤더 Flags 값 중에서 오프셋(offset) 필드의 값을 십진수로 비교할 수 있습니다.

Fragment offset 필드는 분할(Fragmentation)된 패킷을 수신자가 재조합(Reassembly)할 때 패킷의 순서를 확인하는데 사용합니다.

입력 형식

fragoffset:[!|<|>]<number>;



```
alert ip any any -> any any (msg:"IP Header fragoffset keyword"; fragoffset:!0; sid:1000004;)
```

룰 옵션 : ttl

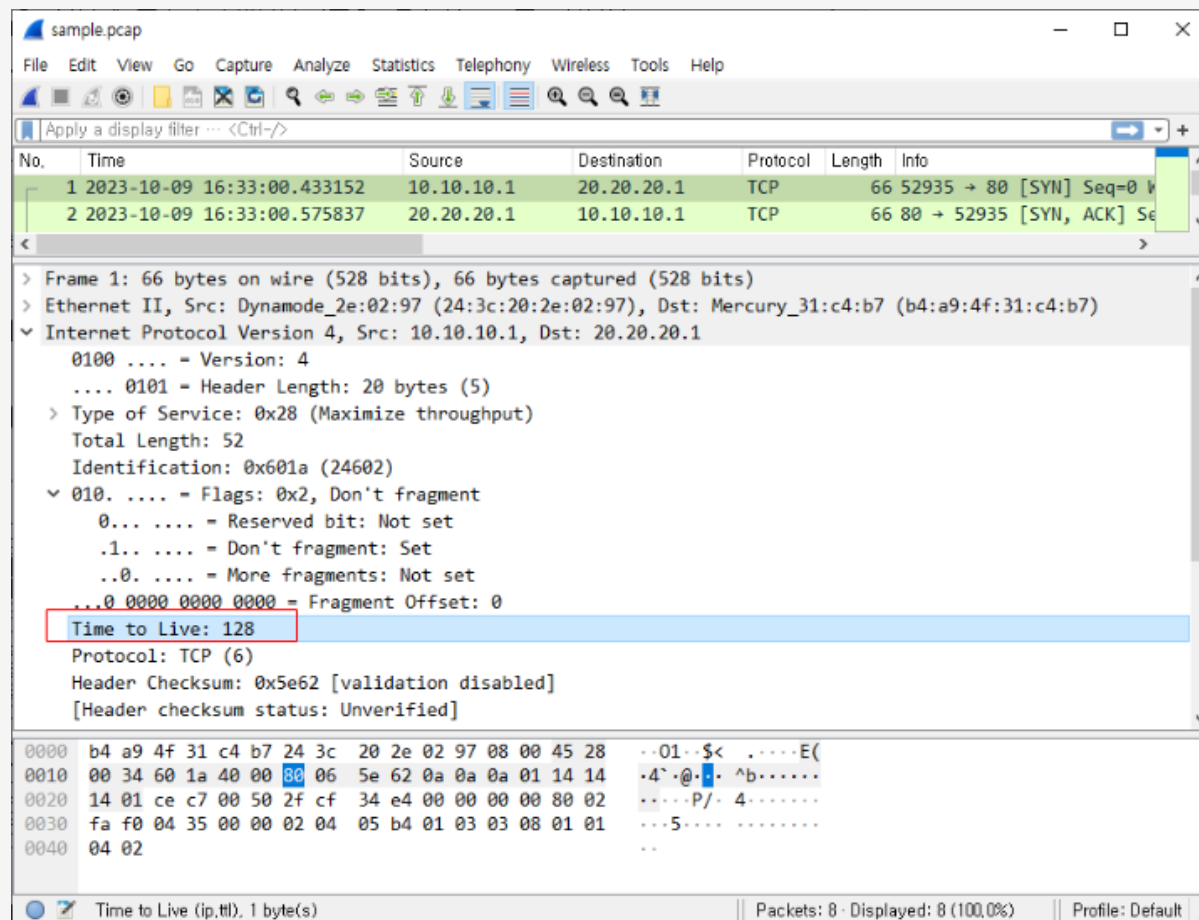
ttl 키워드는 IP 헤더의 Time to Live(TTL) 값을 확인하는데 사용합니다.

TTL 필드는 패킷이 네트워크 내에 너무 오래 머무르는 것을 방지하는 역할을 합니다. 라우터는 패킷이 지나가면 TTL 값을 1씩 감소시켜서 패킷의 TTL 값이 0이 되면, 그 패킷을 폐기시켜서 네트워크 자원의 낭비를 막아주는 역할을 합니다

■ 입력 형식

ttl:[<,>,<=>=<number>;

ttl:[<number>]-[<number>;



```
alert ip any any -> any any (msg:"IP Header ttl keyword"; ttl:43; sid:1000005;)
```

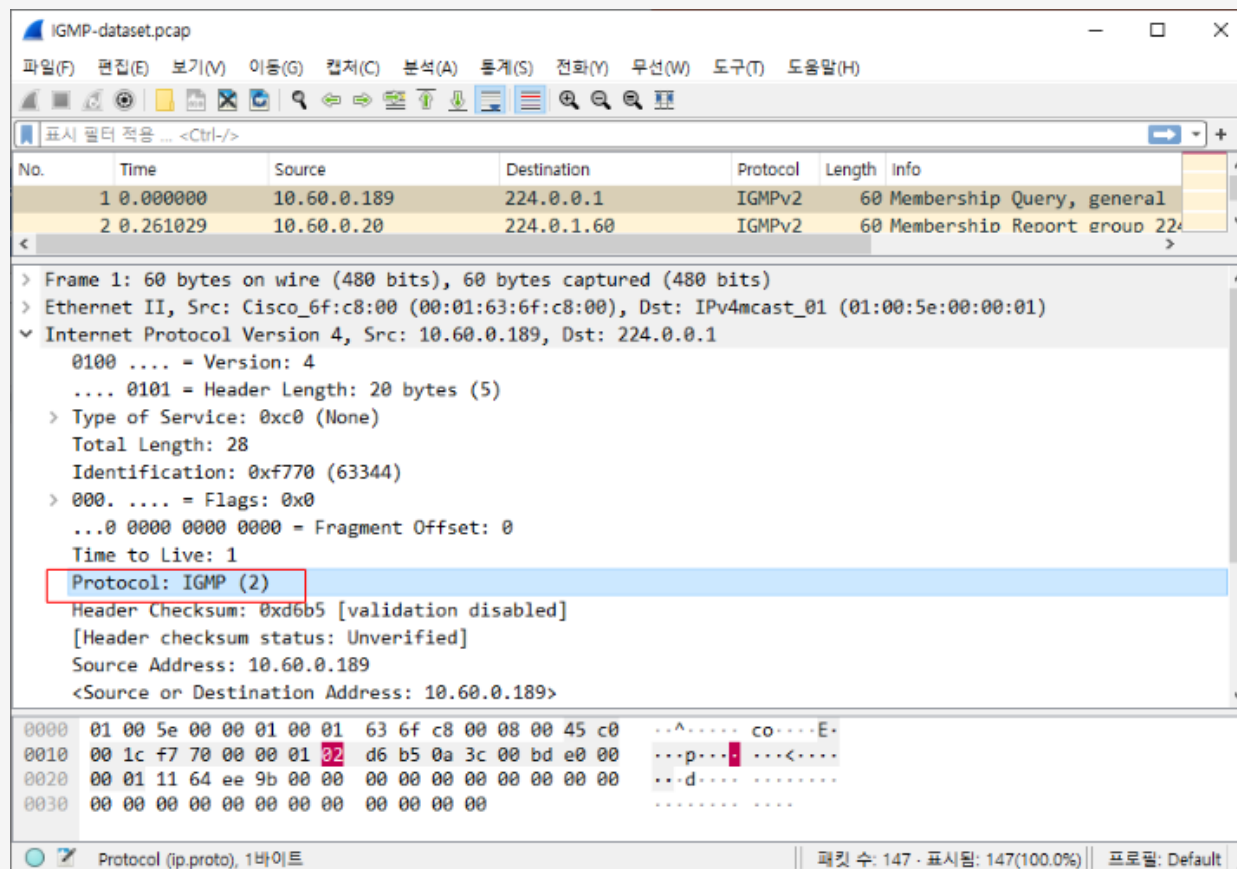
룰 옵션 : ip_proto

ip_proto 키워드는 IPv4 헤더의 Protocol 값을 확인하는데 사용합니다.

Protocol 필드는 상위 계층인 전송 계층 (Transport layer) 프로토콜에서 사용되는 프로토콜에 대한 정보를 제공합니다. 예를 들어, TCP와 UDP 프로토콜은 각각 6과 17의 값을 가집니다.

입력 형식

ip_proto: [!>|<]<[name|number]>;



```
alert ip any any -> any any (msg:"IP Header ip_proto keyword"; ip_proto:2; sid:1000006;)
```

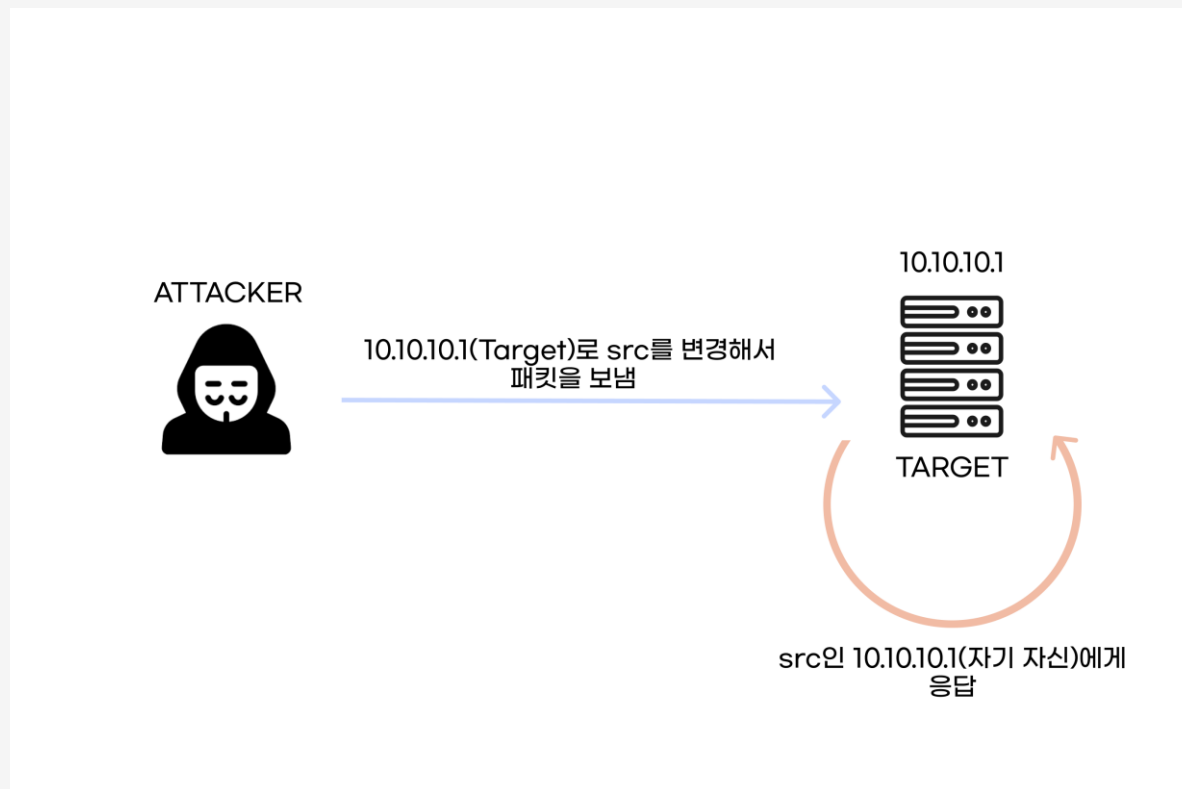
룰 옵션 : sameip

sameip 키워드는 IP 헤더의 Source Address와 Destination Address 값이 같은 지 확인하는데 사용합니다.

이 키워드는 랜드 어택(Land Attack)을 탐지하기 위해 개발되었으나 현재는 대부분의 운영체제에서 이 취약점이 해결되어 이론적으로만 존재하는 공격입니다.

■ 입력 형식

sameip;



```
alert icmp any any -> any any (msg:"IP Header sameip keyword"; sameip; sid:1000007;)
```

룰 옵션 : IP 헤더 구조

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7								
Version				Header Length				Type of Service (tos)								Total Length																							
Identification (id)																Flags (flagbits)		Fragment Offset (flagoffset)																					
Time to Live (ttl)								Protocol (ip_proto)								Header Checksum																							
Source Address																																							
Destination Address																																							

룰 옵션 : TCP 헤더 구조

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Source Port																Destination Port															
Sequence Number																															
Acknowledgment Number																															
Offset				Reserved				TCP Flags								Window															
Checksum																Urgent Pointer															
TCP Options																															

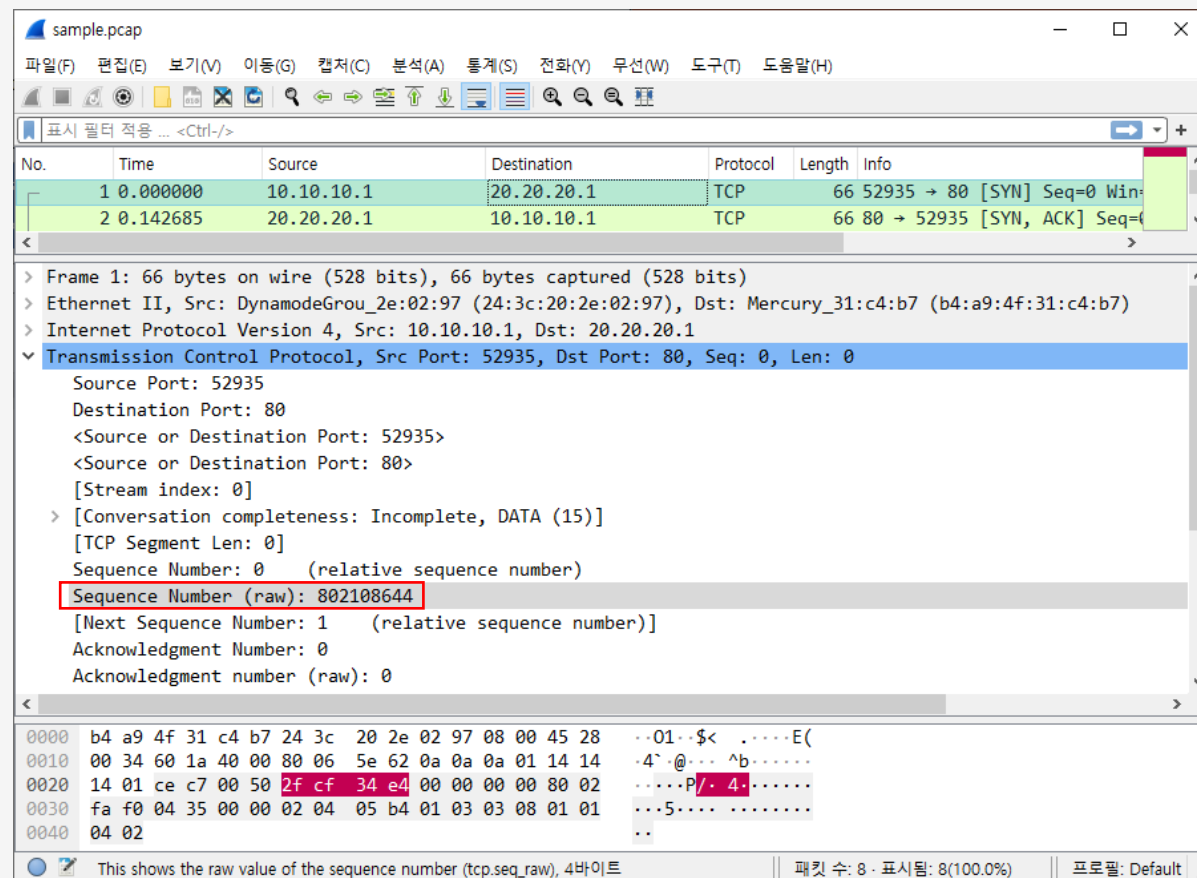
룰 옵션 : seq

seq 키워드는 TCP 헤더의 Sequence Number 값을 확인하는데 사용합니다.

Sequence Number는 TCP 세션이 연결된 동안 얼마나 많은 데이터가 전송되었는지를 나타내는 필드입니다. 예를 들어, 어떤 패킷의 Sequence Number가 X이고, 패킷의 길이가 Y할 때, 이 패킷이 목적지에 성공적으로 전송되면 다음 패킷의 Sequence Number는 X+Y가 됩니다.

■ 입력 형식

seq:<number>;



```
alert tcp any any -> any any (msg:"TCP Header seq keyword"; seq:4092564954; sid:1000008;)
```


룰 옵션 : ack

ack 키워드는 TCP 헤더의 Acknowledge Number 값을 확인하는데 사용합니다.

Sequence Number의 확인 응답으로써, 송신자가 보낸 데이터를 오류 없이 수신했을 경우, 해당 데이터와 함께 수신한 Sequence Number에 1을 더하여 해당 데이터를 정확히 수신했음을 송신자에게 알려줄 때 사용됩니다.

■ 입력 형식

ack:<number>;

sample.pcap

파일(F) 편집(E) 보기(V) 이동(G) 캡처(C) 분석(A) 통계(S) 전화(Y) 무선(W) 도구(T) 도움말(H)

표시 필터 적용 ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.10.1	20.20.20.1	TCP	66	52935 → 80 [SYN] Seq=0 Win=0
2	0.142685	20.20.20.1	10.10.10.1	TCP	66	80 → 52935 [SYN, ACK] Seq=802108645

Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)

Ethernet II, Src: Mercury_31:c4:b7 (b4:a9:4f:31:c4:b7), Dst: DynamodeGrou_2e:02:97 (24:3c:20:2e:02:97)

Internet Protocol Version 4, Src: 20.20.20.1, Dst: 10.10.10.1

Transmission Control Protocol, Src Port: 80, Dst Port: 52935, Seq: 0, Ack: 1, Len: 0

Source Port: 80

Destination Port: 52935

<Source or Destination Port: 80>

<Source or Destination Port: 52935>

[Stream index: 0]

[Conversation completeness: Incomplete, DATA (15)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 3412179957

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 802108645

0000 24 3c 20 2e 02 97 b4 a9 4f 31 c4 b7 08 00 45 00 \$< 01 E .

0010 00 34 00 00 40 00 2b 06 13 a5 14 14 14 01 0a 0a . 4 . . @ . +

0020 0a 01 00 50 ce c7 cb 61 bb f5 2f cf 34 e5 80 12 . . . P . . . a . . / . 4 . .

0030 f5 07 82 b6 00 00 02 04 05 b4 01 01 04 02 01 03

0040 03 07

This shows the raw value of the acknowledgment number (tcp.ack_raw), 4바이트 || 패킷 수: 8 · 표시됨: 8(100.0%) || 프로파일: Default

```
alert tcp any any -> any any (msg:"TCP Header ack keyword"; ack:3076266292;; sid:1000009;)
```

룰 옵션 : flags

flags 키워드는 TCP 헤더의 Flags 값을 확인하는 데 사용됩니다.

TCP Flags는 TCP 연결의 다양한 단계와 상태를 제어하고 관리하는데 사용됩니다.

■ 입력 형식

flags:[!|*|+]<FSRPAUCE0>[,<FSRPAUEC>];

sample.pcap

파일(F) 편집(E) 보기(V) 이동(G) 캡처(C) 분석(A) 통계(S) 전화(Y) 무선(W) 도구(T) 도움말(H)

표시 필터 적용 ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.10.1	20.20.20.1	TCP	66	52935 → 80 [SYN] Seq=0 Win=
2	0.142685	20.20.20.1	10.10.10.1	TCP	66	80 → 52935 [SYN, ACK] Seq=

Sequence Number: 0 (relative sequence number)
 Sequence Number (raw): 3412179957
 [Next Sequence Number: 1 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 802108645
 1000 = Header Length: 32 bytes (8)
 Flags: 0x012 (SYN, ACK)
 000. = Reserved: Not set
 ...0 = Accurate ECN: Not set
0... = Congestion Window Reduced: Not set
0... = ECN-Echo: Not set
0... = Urgent: Not set
1... = Acknowledgment: Set
0... = Push: Not set
0... = Reset: Not set
1... = Syn: Set

0000 24 3c 20 2e 02 97 b4 a9 4f 31 c4 b7 08 00 45 00 \$< 01 E-
 0010 00 34 00 00 40 00 2b 06 13 a5 14 14 14 01 0a 0a -4-@+
 0020 0a 01 00 50 ce c7 cb 61 bb f5 2f cf 34 e5 80 12 . . . P . . . a . . / . 4 . .
 0030 f5 07 82 b6 00 00 02 04 05 b4 01 01 04 02 01 03
 0040 03 07

This shows the raw value of the acknowledgment number (tcp.ack_raw), 4바이트 패킷 수: 8 · 표시됨: 8(100.0%) 프로필: Default

alert tcp any any -> any any (msg:"TCP Header flags keyword"; flags:PA; sid:1000010;)

룰 옵션 : window

window 키워드는 TCP 헤더의 Window 값을 확인하는데 사용합니다.

Window 필드는 수신자가 얼마나 많은 데이터를 받을 수 있는지를 나타냅니다. 기본적으로, TCP 3way-Handshake를 하는 과정에서 자신이 수신할 수 있는 버퍼의 크기가 얼마인지를 알려주고, 네트워크 상황에 따라 변경될 수 있습니다.

■ 입력 형식

window:[!]<number>;

The screenshot shows a Wireshark window titled 'sample.pcap'. The packet list pane shows two packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.10.10.1	20.20.20.1	TCP	66	52935 → 80 [SYN] Seq=0 Win=
2	0.142685	20.20.20.1	10.10.10.1	TCP	66	80 → 52935 [SYN, ACK] Seq=

The packet details pane shows the selected packet (packet 2) with the following fields:

- Sequence Number: 0 (relative sequence number)
- Sequence Number (raw): 3412179957
- [Next Sequence Number: 1 (relative sequence number)]
- Acknowledgment Number: 1 (relative ack number)
- Acknowledgment number (raw): 802108645
- 1000 = Header Length: 32 bytes (8)
- Flags: 0x012 (SYN, ACK)
 - 000. = Reserved: Not set
 - ...0 = Accurate ECN: Not set
 - ... 0... = Congestion Window Reduced: Not set
 - ... 0.. = ECN-Echo: Not set
 -0. = Urgent: Not set
 -1. = Acknowledgment: Set
 -0.. = Push: Not set
 -0.. = Reset: Not set
 - >1. = Syn: Set

The packet bytes pane shows the raw data of the acknowledgment number field (tcp.ack_raw) in hexadecimal and ASCII format.

```
alert tcp any any -> any any (msg:"TCP Header window keyword"; window:502; sid:1000011;)
```

룰 옵션 : TCP 헤더 구조

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Source Port																Destination Port															
Sequence Number (seq)																															
Acknowledgment Number (ack)																															
Offset				Reserved				TCP Flags (flags)								Window (window)															
Checksum																Urgent Pointer															
TCP Options																															

룰 옵션 : ICMP 헤더 구조

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Type								Code								Checksum															
가변 길이 (Type과 Code에 따라 변함)																															

Type이 8이고 Code가 0인 ICMP Echo Request(핑 요청)과 Type이 0이고 Code가 0인 ICMP Echo Reply(핑 응답)은 아래와 같이 Identifier와 Sequence Number 필드가 있습니다.

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Type								Code								Checksum															
Identifier																Sequence Number															

룰 옵션 : itype

itype 키워드는 ICMP 헤더의 Type 값을 확인하는 데 사용됩니다.

ICMP 헤더의 Type 필드는 ICMP 메시지의 종류를 식별하는 데 사용되며, 메시지가 어떤 목적으로 사용되는지를 간략하게 설명하여 수신한 네트워크 장치가 왜 메시지를 받았는지와 어떻게 처리해야 하는지를 알 수 있게 합니다.

■ 입력 형식

itype:[<|>]<number>;

itype:min< > max;

The image shows a Wireshark packet capture window titled 'icmp_sample.pcap'. The packet list pane shows two packets: a ping request (No. 1) and a ping reply (No. 2). The packet details pane for packet 1 is expanded, showing the 'Internet Control Message Protocol' section with 'Type: 8 (Echo (ping) request)' highlighted. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.1.1.1	2.2.2.2	ICMP	86	Echo (ping) request id=0xa005
2	0.000306	2.2.2.2	1.1.1.1	ICMP	86	Echo (ping) reply id=0xa005

Packet 1 details:

- Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
- Ethernet II, Src: Cisco_2b:76:60 (88:f0:77:2b:76:60), Dst: Dell_78:b0:5b (70:b5:e8:78:b0:5b)
- Internet Protocol Version 4, Src: 1.1.1.1, Dst: 2.2.2.2
- Internet Control Message Protocol
 - Type: 8 (Echo (ping) request)
 - Code: 0
 - Checksum: 0x977a [correct] [Checksum Status: Good]
 - Identifier (BE): 40965 (0xa005)
 - Identifier (LE): 1440 (0x05a0)
 - Sequence Number (BE): 41229 (0xa10d)
 - Sequence Number (LE): 3489 (0x0da1)
 - [Response frame: 2]
- Data (44 bytes)

Packet bytes:

```

0000  70 b5 e8 78 b0 5b 88 f0 77 2b 76 60 08 00 45 00  p...x...w+v...E.
0010  00 48 7e 6c 00 00 73 01 c3 43 01 01 01 02 02    .H~1..s..C.....
0020  02 02 08 00 97 7a a0 05 a1 0d 0c fd be 20 00 00  ..z.....
0030  00 00 00 00 00 00 45 45 45 45 45 45 45 45 45 45  ....EE EEEEEEE
0040  45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEE EEEEEEE
0050  45 45 45 45 45 45                                     EEEEE
  
```

```
alert icmp any any -> any any (msg:"ICMP Header itype keyword"; itype:8; sid:1000012;)
```

룰 옵션 : icode

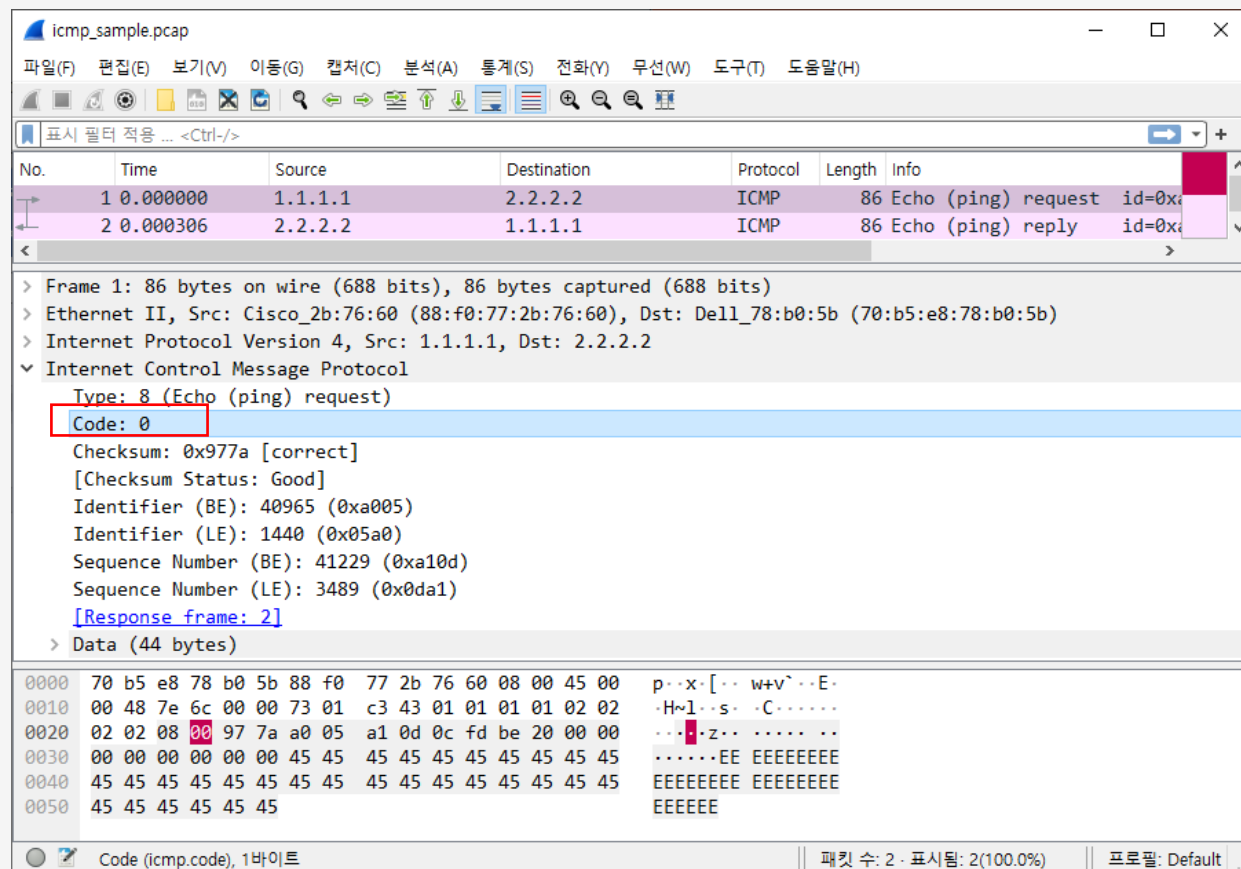
icode 키워드는 ICMP 헤더의 Code 값을 확인하는데 사용합니다.

ICMP 헤더의 Code 필드는 ICMP 메시지의 세부 유형을 나타냅니다. 예를 들어, '목적지 도달 불가'라는 ICMP 메시지 유형(Type 3)에는 16개의 Code가 있습니다.

입력 형식

icode:[<|>]<number>;

icode:min<>max;



```
alert icmp any any -> any any (msg:"ICMP Header icode keyword"; icode:0; sid:1000013;)
```

룰 옵션 : icmp_id

icmp_id 키워드는 ICMP 헤더의 Identifier 값을 확인하는데 사용합니다.

ICMP 헤더의 Identifier는 ICMP 메시지를 식별하는 데 사용됩니다. ICMP Echo Request(핑 요청)과 ICMP Echo Reply(핑 응답)에서 TCP/UDP 포트와 같이 ICMP 세션을 식별할 수 있습니다.

입력 형식

icode:[<|>]<number>;

icode:min<>max;

icmph_sample.pcap

파일(F) 편집(E) 보기(V) 이동(G) 캡처(C) 분석(A) 통계(S) 전화(Y) 무선(W) 도구(T) 도움말(H)

표시 필터 적용 ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.1.1.1	2.2.2.2	ICMP	86	Echo (ping) request id=0xa005
2	0.000306	2.2.2.2	1.1.1.1	ICMP	86	Echo (ping) reply id=0xa005

> Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)

> Ethernet II, Src: Cisco_2b:76:60 (88:f0:77:2b:76:60), Dst: Dell_78:b0:5b (70:b5:e8:78:b0:5b)

> Internet Protocol Version 4, Src: 1.1.1.1, Dst: 2.2.2.2

> Internet Control Message Protocol

Type: 8 (Echo (ping) request)

Code: 0

Checksum: 0x977a [correct]

[Checksum Status: Good]

Identifier (BE): 40965 (0xa005)

Identifier (LE): 1440 (0x05a0)

Sequence Number (BE): 41229 (0xa10d)

Sequence Number (LE): 3489 (0x0da1)

[Response frame: 2]

> Data (44 bytes)

```

0000  70 b5 e8 78 b0 5b 88 f0 77 2b 76 60 08 00 45 00  p...x[...w+v^...E
0010  00 48 7e 6c 00 00 73 01 c3 43 01 01 01 01 02 02  .H~l...s...C.....
0020  02 02 08 00 97 7a a0 05 a1 0d 0c fd be 20 00 00  ....z...
0030  00 00 00 00 00 00 45 45 45 45 45 45 45 45 45 45  ....EE EEEEEEEE
0040  45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEE EEEEEEEE
0050  45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45  EEEEEEEE
  
```

Identifier (big endian representation) (icmp.ident), 2바이트

패킷 수: 2 · 표시됨: 2(100.0%)

프로필: Default

```
alert icmp any any -> any any (msg:"ICMP Header icmp_id keyword"; icmp_id:40965; sid:1000014;)
```


룰 옵션 : icmp_seq

icmp_seq 키워드는 ICMP 헤더의 Sequence Number 값을 확인하는데 사용합니다.

ICMP 헤더의 Sequence Number는 패킷의 순서를 나타냅니다. 이 값은 각각의 ICMP Echo Request에 대해 1씩 증가합니다. 이를 통해 어떤 Echo Request에 대한 Echo Reply인지를 알 수 있습니다.

■ 입력 형식

icmp_seq:<number>;

Sequence Number (big endian representation) (icmp.seq), 2바이트

패킷 수: 2 · 표시됨: 2(100.0%)

프로필: Default

```
alert icmp any any -> any any (msg:"ICMP Header icmp_seq keyword"; icmp_seq:41229; sid:1000015;)
```

룰 옵션 : ICMP 헤더 구조

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Type (itype)								Code (icode)								Checksum															
Identifier (icmp_id)																Sequence Number (icmp_seq)															

Part III. 패킷 페이로드 검사하기 (content)

룰 옵션 : content

SNORT는 패킷의 페이로드 영역에서 문자열을 검사할 수 있는 다양한 키워드를 제공합니다.

페이로드(Payload)는 패킷 헤더를 제외한 데이터 영역을 말합니다. 일반적으로 취약점, 악성코드 등을 탐지할 때, 이 영역에서 시그니처를 추출하여 패턴 매칭을 합니다.

The image shows a Wireshark packet capture of a network traffic. The top pane shows a list of packets, with packet 4 selected. The middle pane shows the details of packet 4, which is an HTTP GET request. The bottom pane shows the raw packet data in hexadecimal and ASCII. The ASCII view highlights the payload of the HTTP request, which is a SQL injection attack.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.30.1.20	172.30.1.100	TCP	74	33558 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=
2	0.000100	172.30.1.100	172.30.1.20	TCP	66	80 → 33558 [SYN, ACK] Seq=0 Ack=1 Win=6553
3	0.001536	172.30.1.20	172.30.1.100	TCP	60	33558 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=
4	0.001894	172.30.1.20	172.30.1.100	HTTP	395	GET /glpi/scripts/unlock_tasks.php?cycle=1
5	0.008422	172.30.1.100	172.30.1.20	TCP	238	80 → 33558 [PSH, ACK] Seq=1 Ack=342 Win=65

Frame 4: 395 bytes on wire (3160 bits), 395 bytes captured (3160 bits)

Ethernet II, Src: DynamodeGrou_2e:02:97 (24:3c:20:2e:02:97), Dst: DynamodeGrou_2e:02:97 (24:3c:20:2e:02:97)

Internet Protocol Version 4, Src: 172.30.1.20, Dst: 172.30.1.100

Transmission Control Protocol, Src Port: 33558, Dst Port: 80, Seq: 1, Ack: 1, Len: 341

Hypertext Transfer Protocol

```

0000  24 3c 20 2e 02 97 24 3c 20 2e 02 97 08 00 45 00  $<...$<.....E-
0010  01 7d 0c b3 40 00 40 06 d2 13 ac 1e 01 14 ac 1e  .}..@.@.....
0020  01 64 83 16 00 50 f7 8d 4b 23 1c 8e e4 b5 50 18  -d...P...K#...P.
0030  01 f6 d3 03 00 00 47 45 54 20 2f 67 6c 70 69 2f  ....GE T /glpi/
0040  73 63 72 69 70 74 73 2f 75 6e 6c 6f 63 6b 5f 74  scrip.../ unlock t
0050  61 73 6b 73 2e 70 68 70 3f 63 79 63 6c 65 3d 31  asks.php ?cycle=1
0060  25 32 30 55 4e 49 4f 4e 25 32 30 41 4c 4c 25 32  %20UNION %20ALL%2
0070  30 53 45 4c 45 43 54 25 32 30 31 2c 28 40 40 76  0SELECT% 201,(@@v
0080  65 72 73 69 6f 6e 29 2d 2d 25 32 30 26 6f 6e 6c  ersion)- %20&onl
0090  79 5f 74 61 73 6b 73 3d 31 20 48 54 54 50 2f 31  y_tasks= 1 HTTP/1
00a0  2e 31 0d 0a 48 6f 73 74 3a 20 31 37 32 2e 33 30  .1..Host : 172.30
00b0  2e 31 2e 31 30 30 0d 0a 55 73 65 72 2d 41 67 65  .1.100.. User-Age
00c0  6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20  nt: Mozilla/5.0
00d0  28 4d 61 63 69 6e 74 6f 73 68 3b 20 49 6e 74 65  (Macintosh; Inte
00e0  6c 20 4d 61 63 20 4f 53 20 58 20 31 30 5f 39 5f  l Mac OS X 10_9_
00f0  33 29 20 41 70 70 6c 65 57 65 62 4b 69 74 2f 35  3) AppleWebKit/5
0100  33 37 2e 33 36 20 28 4b 48 54 4d 4c 2c 20 6c 69  37.36 (KHTML, li
0110  6b 65 20 47 65 63 6b 6f 29 20 43 68 72 6f 6d 65  ke Gecko ) Chrome
0120  2f 33 35 2e 30 2e 31 39 31 36 2e 34 37 20 53 61  /35.0.19 16.47 Sa
0130  66 61 72 69 2f 35 33 37 2e 33 36 0d 0a 43 6f 6e  fari/537 .36..Con
0140  6e 65 63 74 69 6f 6e 3a 20 63 6c 6f 73 65 0d 0a  nection: close..
0150  41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 41 63 63  Accept: /*.*.Acc
0160  65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e  ept-Lang uage: en
0170  0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e  ..Accept -Encodin
0180  67 3a 20 67 7a 69 70 0d 0a 0d 0a                g: gzip: ...
  
```

Hypertext Transfer Protocol (http), 341 바이트

패킷 수: 12 · 표시됨: 12(100.0%)

프로필: Default

룰 옵션 : content

여러분이 해당 취약점을 탐지할 수 있는 시그니처를 만들어 보세요.

(조건) content 키워드를 3개 이상 사용하세요.

[별첨1. 취약점 정보](#)

- **입력 형식** : content:[!]"[<text>|<binary>]";
- **사용 예제** : ①content:"ABCabc"; ②content:"|31 33 33|"; ③content:"AaBb|31 32 33|";

- **주의 할 점**

SNORT에서 구분자로 사용하는 문자를 탐지 문자열로 입력할 때는 주의가 필요합니다.

- 쌍따옴표(") : 입력한 문자열의 시작과 끝을 구분하는데 사용
- 파이프(|) : 바이너리 데이터의 시작과 끝을 구분하는데 사용
- 세미콜론(;) : 모든 키워드의 끝을 구분하는데 사용

```
alert tcp any any -> any 80 (msg:"content keyword"; content:"input strings"; sid:1000016;)
```

PCAP 샘플을 변경 후 다시 한번 탐지 테스트를 해보세요.

- 탐지가 잘 되었나요?
- 탐지가 왜 안되었을까요?
- 이러한 문제를 어떻게 해결하면 좋을까요?

룰 옵션 : nocase

sample_sql_i_evasion.pcap

nocase 키워드는 content의 modifier로 탐지 문자열을 대소문자 구분 없이 매칭할 수 있게 해줍니다.
공격자는 보안 솔루션을 우회하기 위해서 공격 문자열에 대소문자를 섞어서 사용하는데,
이러한 우회 공격을 탐지할 때 사용합니다.

- **입력 형식** : nocase;
- **사용 예제** : ①content:"AbC"; nocase; ②content:"|41 62 43|"; nocase;

여러분이 작성한 content에 nocase를 추가하고, 다시 한번 탐지 테스트를 해보세요.

룰 옵션 : offset

offset 키워드는 패턴 매칭을 시작 할 위치를 지정하는데 사용합니다.

일반적으로 depth 키워드와 함께 사용하며, 탐색 범위를 제한하여 정탐을 높이는데 활용합니다.

offset 값은 숫자로 입력할 수도 있지만, byte_extract 키워드에서 지정한 변수명으로 입력할 수도 있습니다.

offset 키워드를 사용하지 않으면, 페이로드의 첫번째 바이트부터 패턴 매칭을 수행합니다.

- **입력 형식** : offset:[<number>|<var_name>];
- **사용 예제** : content:"GET"; offset:0;

offset 예제 (1/2)

[Rule]

alert tcp any any -> any any (msg:"offset Test"; content:"EXEC"; **offset:4;**)

[Packet]



53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76
69 6C 66 6F 6F 0A

SITE EXEC ev
ilfoo.

offset 예제 (2/2)

[Rule]

```
alert tcp any any -> any any (msg:"offset Test"; content:"EXEC"; offset:4;)
```

[Packet]

53 49 54 45 20 20 20 20 20 20 45 58 45 43 20 65 76
69 6C 66 6F 6F 0A



SITE EXEC ev
ilfoo.

룰 옵션 : depth

depth 키워드는 패턴 매칭의 종료 위치를 지정하는데 사용합니다.

일반적으로 offset 키워드와 함께 사용하며, 탐색 범위를 제한하여 정탐을 높이는데 활용합니다.

depth 값은 숫자로 입력할 수도 있지만, byte_extract 키워드에서 지정한 변수명으로 입력할 수도 있습니다.

- **입력 형식** : depth:[<number>|<var_name>];
- **사용 예제** : content:"@@version"; offset:10; depth:9;

depth 예제 (1/3)

[Rule]

alert tcp any any -> any any (msg:"offset Test"; content:"EXEC"; **offset:4**; depth:10;)

[Packet]



53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76
69 6C 66 6F 6F 0A


SITE EXEC ev
ilfoo.


depth 예제 (2/3)

[Rule]

alert tcp any any -> any any (msg:"offset Test"; content:"EXEC"; offset:4; depth:10;)

[Packet]

 53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76
69 6C 66 6F 6F 0A

 SITE EXEC ev
ilfoo.

depth 예제 (3/3)

[Rule]

alert tcp any any -> any any (msg:"offset Test"; **content:"EXEC"**; offset:4; depth:10;)

[Packet]

53 49 54 45 20 20 20 20 20 20 **45 58 45 43** 20 65 76
69 6C 66 6F 6F 0A



SITE EXEC ev
ilfoo.

실습 : offset과 depth

sample_sqli.pcap

여러분이 작성한 룰에 [보기]의 문자열을 탐지하는 content를 추가하고,
정확한 offset과 depth 값을 찾아보세요.

[보기]

content:"GET"; offset: ; depth: ;

룰 옵션 : distance

distance 키워드는 두 번째 content 부터 사용할 수 있으며, 이전 content에 매칭된 문자열의 끝부터 얼마만큼의 거리를 두고 문자열을 매칭할 지 설정할 수 있습니다.

단독으로도 많이 사용하지만 within과 함께 사용할 경우 패턴 매칭 범위를 특정할 수 있습니다.

음수(-) 값을 넣어서 이전에 탐지된 문자열의 앞으로 이동하여 패턴 매칭을 할 수도 있습니다.

- **입력 형식** : distance:[-][<number>|<var_name>];
- **사용 예제** : content:"GET"; content:"cycle="; distance:0;

distance 예제 (1/3)

[Rule]

alert tcp any any -> any any (msg:"Distance Test"; **content:"SITE"**; content:"EXEC"; distance:0;)

[Packet]



53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76 SITE EXEC ev
69 6C 66 6F 6F 0A ilfoo.

distance 예제 (2/3)

[Rule]

alert tcp any any -> any any (msg:"Distance Test"; content:"SITE"; content:"EXEC"; **distance:1;**)

[Packet]


53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76 SITE EXEC ev
69 6C 66 6F 6F 0A ilfoo.

distance 예제 (3/3)

[Rule]

alert tcp any any -> any any (msg:"Distance Test"; content:"SITE"; **content:"EXEC"**; distance:0;)

[Packet]

53 49 54 45 20 20 20 20 20 20 **45 58 45 43** 20 65 76
69 6C 66 6F 6F 0A



SITE EXEC ev
ilfoo.

룰 옵션 : within

within 키워드는 두 번째 content 부터 사용할 수 있으며, 이전 content에 매치된 문자열 이후부터 지정한 범위 안에서 문자열을 매칭하는데 사용합니다.

- **입력 형식** : within:[<number>|<var_name>];
- **사용 예제** : content:"GET"; content:"cycle="; distance:0; content:"SELECT"; distance:0; within:50;

within 예제 (1/4)

[Rule]

alert tcp any any -> any any (msg:"Within Test"; **content:"SITE"**; content:"EXEC"; distance:1; within:10;)

[Packet]



53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76
69 6C 66 6F 6F 0A


SITE EXEC ev
ilfoo.

within 예제 (2/4)

[Rule]

alert tcp any any -> any any (msg:"Within Test"; content:"SITE"; content:"EXEC"; **distance:1**; within:10;)

[Packet]



53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76 SITE EXEC ev
69 6C 66 6F 6F 0A ilfoo.

within 예제 (3/4)

[Rule]

alert tcp any any -> any any (msg:"Within Test"; content:"SITE"; content:"EXEC"; distance:1; within:10;)

[Packet]

 53 49 54 45 20 20 20 20 20 45 58 45 43 20 65 76
69 6C 66 6F 6F 0A

 SITE EXEC ev
ilfoo.

within 예제 (4/4)

[Rule]

alert tcp any any -> any any (msg:"Within Test"; content:"SITE"; **content:"EXEC"**; distance:1; within:10;)

[Packet]

53 49 54 45 20 20 20 20 20 20 **45 58 45 43** 20 65 76
69 6C 66 6F 6F 0A



SITE EXEC ev
ilfoo.

실습 : distance와 within

sample_sqli.pcap

여러분이 작성한 룰에 [보기]의 문자열을 탐지하는 content를 추가하고,
정확한 distance와 within 값을 찾아보세요.

[보기]

content:"@@version"; distance: ; within: ;

룰 옵션 : fast_pattern

fast_pattern 키워드는 패턴 매칭의 성능을 높이기 위해서, 가장 먼저 검사 할 수 있는 content를 직접 지정하는데 사용합니다.

이 키워드를 사용하지 않으면 content 키워드 중에서 가장 긴 문자열을 fast_pattern으로 선정하는데, 가장 긴 문자열을 선정하는 이유는 유니크한 문자열이 될 가능성이 높기 때문입니다.

- **입력 형식** : ① fast_pattern; ② fast_pattern:only; ③ fast_pattern:<offset>,<length>;
- **사용 예제** : content:"GET"; content:"cycle="; distance:0; fast_pattern; content:"version"; distance:0;

fast_pattern의 탄생 배경 (1/2)

다음 Snort Rule에서 ①은 룰 헤더로 RTN(Rule Tree Node)라고 하고, ②은 룰 옵션으로 OTN(Option Tree Node)라고 합니다. ①과 ② 중에서 먼저 검사하는 것을 어떤 것일까요?

- ① alert tcp any any -> any 80
- ② (msg:"snort detection"; flow:to_server,established; content:"GET"; offset:0; depth:3; content:"/unlock_tasks.php"; distance:0; content:"cycle="; distance:0; content:"@@version"; distance:0; within:100; sid:1;)

일반적으로 ①RTN을 먼저 검사하고 ②OTN을 검사할 것이라고 생각하기 쉽습니다.

①RTN을 먼저 검사할 경우, 어떤 일이 생길까요?

룰이 1만개 있다고 가정해 봅시다.

하나의 패킷이 들어오면 가지고 있는 룰에서 일치하는 포트가 있는지 찾아야 합니다.

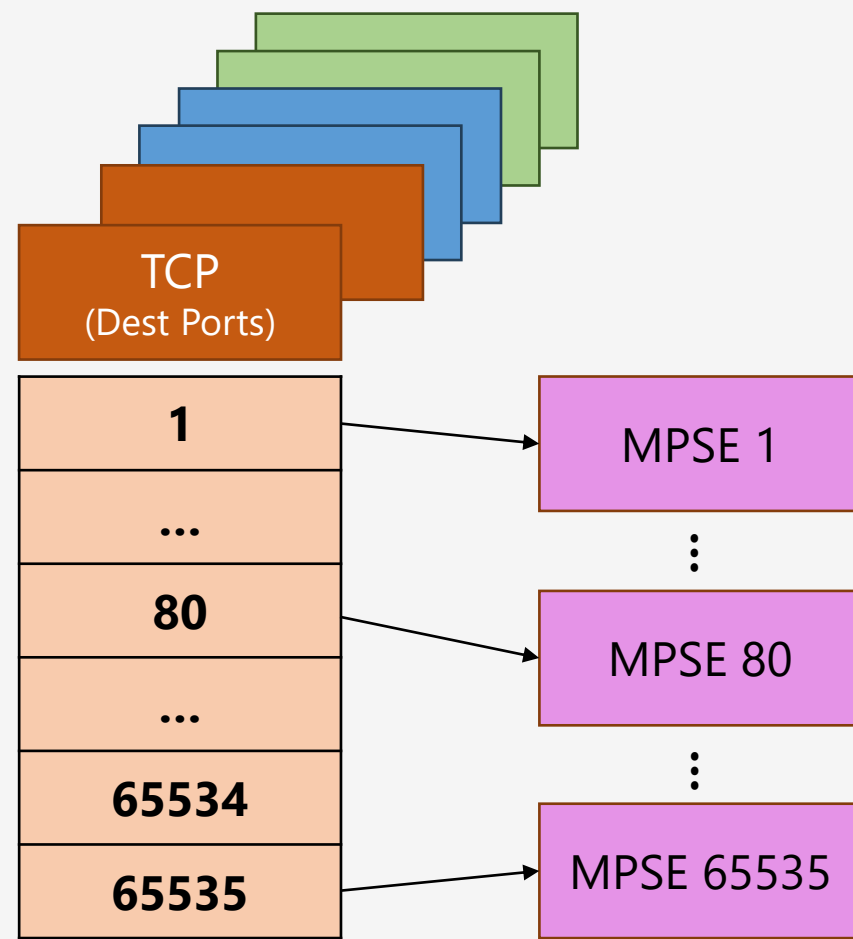
이 작업을 1만 번 한다고 생각해 보세요. 룰이 증가할 수록 처리 속도는 점점 감소할 것입니다.

fast_pattern의 탄생 배경 (2/2)

- ① alert **tcp** any any -> any **80**
- ② (msg:"snort detection"; flow:to_server,established;
- ③ content:"**GET**"; offset:0; depth:3;
- ④ content:"**/unlock_tasks.php**"; distance:0;
- ⑤ content:"**cycle=**"; distance:0;
- ⑥ content:"**@@version**"; distance:0; within:100; **fast_pattern**;
- ⑦ sid:1;)

```
GET /glpi/scripts/unlock_tasks.php?  
cycle=1%20UNION%20ALL%20SELECT%201,(@@version)--  
%20&only_tasks=1  
HTTP/1.1
```

fast_pattern 관리 방법



룰 옵션 : http_method

http_method 키워드는 문자열 검색 범위를 HTTP 메서드로 제한합니다.

HTTP 메서드의 종류에는 GET, POST, PUT, HEAD, OPTIONS, DELETE, TRACE 등이 있으며, 특정 HTTP 메서드를 사용하는 트래픽을 탐지하는데 사용할 수 있습니다.

- **입력 형식** : http_method;
- **사용 예제** : content:"GET"; http_method;

```
GET /board/index.php?id=123 HTTP/1.1
```

```
Host: www.server.com
```

```
User-Agent: Mozilla/5.0 Chrome
```

```
Connection: close
```

```
Accept: */*
```

```
Accept-Language: en
```

```
Accept-Encoding: gzip
```

```
Cookie: PHPSESSID=j3vdo1c4o1u;
```

```
alert tcp any any -> any 80 (msg:"http_method keyword"; content:"GET"; http_method; sid:1000100;)
```

룰 옵션 : http_uri 와 http_raw_uri

http_uri 와 http_raw_uri 키워드는 URI 필드에서 문자열을 검색하는데 사용합니다.

두 키워드의 차이점은 URI 필드의 [정규화\(Normalization\)](#) 처리 여부로, http_uri는 정규화 된 URI 데이터에서 문자열 검색을 할 수 있습니다.

- **입력 형식** : ①http_uri; ②http_raw_uri;
- **사용 예제** : content:"/unlock_tasks.php"; http_uri;

```
GET /board/index.php?id=123 HTTP/1.1
```

```
Host: www.server.com
```

```
User-Agent: Mozilla/5.0 Chrome
```

```
Connection: close
```

```
Accept: */*
```

```
Accept-Language: en
```

```
Accept-Encoding: gzip
```

```
Cookie: PHPSESSID=j3vdo1c4o1u;
```

```
alert tcp any any -> any 80 (msg:"http_uri keyword"; content:"UNION ALL SELECT"; http_uri;  
sid:1000101;)
```

룰 옵션 : http_header 와 http_raw_header

http_header와 http_raw_header 키워드는 HTTP 요청과 응답의 헤더 필드에서 문자열을 검색하는데 사용합니다.

두 키워드의 차이점은 HTTP 헤더 필드의 정규화 (Normalization) 처리 여부입니다.

- **입력 형식** : ①http_header; ②http_raw_header;
- **사용 예제** : content:"Host: www.server.com"; http_header;

```
GET /board/index.php?id=123 HTTP/1.1
```

```
Host: www.server.com
```

```
User-Agent: Mozilla/5.0 Chrome
```

```
Connection: close
```

```
Accept: */*
```

```
Accept-Language: en
```

```
Accept-Encoding: gzip
```

```
Cookie: PHPSESSID=j3vdo1c4o1u;
```

```
alert tcp any any -> any 80 (msg:"http_header keyword"; content:"Chrome/35.0.1916.47"; http_header;  
sid:1000102;)
```

룰 옵션 : http_cookie 와 http_raw_cookie

http_cookie와 http_raw_cookie 키워드는 HTTP 요청과 응답의 Cookie 헤더 필드에서 문자열을 검색하는데 사용됩니다.

두 키워드의 차이점은 HTTP 헤더 필드의 정규화(Normalization) 처리 여부입니다.

- **입력 형식** : ①http_cookie; ②http_raw_cookie;
- **사용 예제** : content:"admin"; http_cookie;

```
GET /board/index.php?id=123 HTTP/1.1
Host: www.server.com
User-Agent: Mozilla/5.0 Chrome
Connection: close
Accept: */*
Accept-Language: en
Accept-Encoding: gzip
Cookie: PHPSESSID=adminj3vdo1c4o1u;
```

```
alert tcp any any -> any 80 (msg:"http_cookie keyword"; content:"WP+Cookie+check"; http_cookie;
sid:1000103;)
```


룰 옵션 : http_client_body

http_client_body 키워드는 HTTP 요청의 본문(Body) 필드에서 문자열을 검색하는데 사용합니다.

보통은 HTTP 요청 메서드 중에서 본문을 가질 수 있는 POST, PUT, PATCH 등에 사용하며, 매개변수와 입력 값 검사 그리고 업로드 되는 파일 확장자와 파일 내용 등을 검사할 때 사용할 수 있습니다.

또한, 본문 필드는 정규화를 하지 않기 때문에, %xx 형식으로 인코딩 된 문자열을 그대로 패턴 매칭해야 합니다.

- **입력 형식** : http_client_body;
- **사용 예제** : content:"password="; http_client_body;

```
POST /login.php HTTP/1.1
Host: www.server.com
User-Agent: Mozilla/5.0 Chrome
Connection: close
Content-Length: 54
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
```

```
id=root&password=qwe123
```

```
alert tcp any any -> any 80 (msg:"http_client_body"; content:"&wp-submit="; http_client_body;
sid:1000104;)
```

룰 옵션 : http_stat_code

http_stat_code 키워드는 HTTP 서버 응답 코드 필드를 검색하는데 사용합니다.

대표적인 코드표는 다음과 같습니다.

코드	설명	예시
1xx	정보 제공	100 Continue
2xx	성공	200 OK
3xx	리다이렉션	302 Found
4xx	클라이언트 에러	404 Not Found
5xx	서버 에러	500 Internal Server Error

```
HTTP/1.1 200 OK
Server: SimpleHTTP
Date: Sun, 16 Jul 2023 08:48:28 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 469
```

```
<html>
<title>Web Server</title>
```

- 입력 형식 : http_stat_code;
- 사용 예제 : content:"200"; http_stat_code;

```
#alert tcp any 80 -> any any (msg:"http_stat_code"; content:"200"; http_stat_code; sid:1000105;)
```

룰 옵션 : http_stat_msg

http_stat_code 키워드는 HTTP 서버 응답 메시지 필드를 검색하는데 사용합니다.

대표적인 코드표는 다음과 같습니다.

코드	설명	예시
200	OK	서버가 요청을 성공적으로 처리했을 때 발생
302	Found	일시적으로 콘텐츠가 이동했을 때 발생
401	Unauthorized	권한 없음, 인증 없이 서버에 접근했을 때 발생
404	Not Found	요청한 리소스가 없을 때 발생
500	Internal Server Error	서버에 오류가 발생해 작업을 수행할 수 없을 때 발생

```
HTTP/1.1 200 OK
Server: SimpleHTTP
Date: Sun, 16 Jul 2023 08:48:28 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 469
```

```
<html>
<title>Web Server</title>
```

- **입력 형식** : http_stat_msg
- **사용 예제** : content:"OK"; http_stat_msg;

```
alert tcp any 80 -> any any (msg:"http_sta_msg"; content:"OK"; http_stat_msg; sid:1000106;)
```

Part IV. 패킷 페이로드 검사하기 (PCRE)

룰 옵션 : pcre

pcre 키워드는 펄(Perl) 호환 정규식을 사용하여 더 복잡한 탐지 문자열을 매칭하는데 사용합니다. 그러나 pcre는 content에 비해 더 많은 계산 자원이 필요하며, 처리 속도가 6배 이상 느리기 때문에 보통은 content 다음에 pcre 순서로 작성하여 content의 문자열이 먼저 탐지되면 그 다음에 pcre의 문자열이 탐지될 수 있도록 하였습니다.

■ 입력 형식

pcre:[!]"(/<regex>/[ismxAEGRUBPHMCOIDKYS]);

GLPI_9.3.3_SQL_injection.pcap

tcp.stream eq 0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.30.1.20	172.30.1.100	TCP	74	33558 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=
2	0.000100	172.30.1.100	172.30.1.20	TCP	66	80 → 33558 [SYN, ACK] Seq=0 Ack=1 Win=6553
3	0.001536	172.30.1.20	172.30.1.100	TCP	60	33558 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=
4	0.001894	172.30.1.20	172.30.1.100	HTTP	395	GET /glpi/scripts/unlock_tasks.php?cycle=1
5	0.008422	172.30.1.100	172.30.1.20	TCP	238	80 → 33558 [PSH, ACK] Seq=1 Ack=342 Win=65

Frame 4: 395 bytes on wire (3160 bits), 395 bytes captured (3160 bits)

Ethernet II, Src: DynamodeGrou_2e:02:97 (24:3c:20:2e:02:97), Dst: DynamodeGrou_2e:02:97 (24:3c:20:2e:02:97)

Internet Protocol Version 4, Src: 172.30.1.20, Dst: 172.30.1.100

Transmission Control Protocol, Src Port: 33558, Dst Port: 80, Seq: 1, Ack: 1, Len: 341

Hypertext Transfer Protocol

```

0000 24 3c 20 2e 02 97 24 3c 20 2e 02 97 08 00 45 00 $<...$<.....E-
0010 01 7d 0c b3 40 00 40 06 d2 13 ac 1e 01 14 ac 1e .}..@..@.....
0020 01 64 83 16 00 50 f7 8d 4b 23 1c 8e e4 b5 50 18 -d...P...K#....P.
0030 01 f6 d3 03 00 00 47 45 54 20 2f 67 6c 70 69 2f .....GE T /glpi/
0040 73 63 72 69 70 74 73 2f 75 6e 6c 6f 63 6b 5f 74 scrips/ unlock t
0050 61 73 6b 73 2e 70 68 70 3f 63 79 63 6c 65 3d 31 asks.php ?cycle=1
0060 25 32 30 55 4e 49 4f 4e 25 32 30 41 4c 4c 25 32 %20UNION %20ALL%2
0070 30 53 45 4c 45 43 54 25 32 30 31 2c 28 40 40 76 0SELECT% 201,(@@v
0080 65 72 73 69 6f 6e 29 2d 2d 25 32 30 26 6f 6e 6c ersion)- %20&onl
0090 79 5f 74 61 73 6b 73 3d 31 20 48 54 54 50 2f 31 y_tasks= 1 HTTP/1
00a0 2e 31 0d 0a 48 6f 73 74 3a 20 31 37 32 2e 33 30 .1..Host : 172.30
00b0 2e 31 2e 31 30 30 0d 0a 55 73 65 72 2d 41 67 65 .1.100.. User-Age
00c0 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 nt: Mozi lla/5.0
00d0 28 4d 61 63 69 6e 74 6f 73 68 3b 20 49 6e 74 65 (Macinto sh; Inte
00e0 6c 20 4d 61 63 20 4f 53 20 58 20 31 30 5f 39 5f 1 Mac OS X 10.9
00f0 33 29 20 41 70 70 6c 65 57 65 62 4b 69 74 2f 35 3) Apple WebKit/5
0100 33 37 2e 33 36 20 28 4b 48 54 4d 4c 2c 20 6c 69 37.36 (K HTML, li
0110 6b 65 20 47 65 63 6b 6f 29 20 43 68 72 6f 6d 65 ke Gecko ) Chrome
0120 2f 33 35 2e 30 2e 31 39 31 36 2e 34 37 20 53 61 /35.0.19 16.47 Sa
0130 66 61 72 69 2f 35 33 37 2e 33 36 0d 0a 43 6f 6e fari/537 .36..Con
0140 6e 65 63 74 69 6f 6e 3a 20 63 6c 6f 73 65 0d 0a nnection: close..
0150 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 41 63 63 Accept: /*.*Acc
0160 65 70 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e ept-Lang uage: en
0170 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 69 6e ..Accept -Encodin
0180 67 3a 20 67 7a 69 70 0d 0a 0d 0a g: gzip: ...
  
```

Hypertext Transfer Protocol (http), 341바이트

패킷 수: 12 · 표시됨: 12(100.0%)

프로필: Default

룰 옵션 : pcre 문법

일반적인 pcre 문법은 다음과 같습니다.

기호	설명	기호	설명
\d	숫자	{n,m}	앞의 문자가 n에서 m개 사이로 반복
\D	\d (숫자)를 제외한 모든 문자	[abc]	[]안의 문자(abc 단어 아님) 중 1개
\w	알파벳 대소문자와 숫자, 언더바(_)	[0-9]	0~9 사이의 숫자
\W	\w 를 제외한 모든 문자	[a-z]	a ~ z 사이의 소문자
\b	문자와 공백()	[A-Z]	A ~ Z 사이의 대문자
\x	바이너리 문자 표현식	[^문자]	입력한 문자를 제외한 모든 문자
\s	공백 문자	^	문자의 시작
\t	탭 문자	\$	문자의 끝
\r	캐리지 리턴 문자	?	앞의 문자가 0개 또는 1개
\n	줄바꿈 문자	*	앞의 문자가 0개 이상 반복
\	이스케이프 문자	+	앞의 문자가 1개 이상 반복
{n}	앞의 문자가 n개 반복	.	줄바꿈 문자를 제외한 모든 문자
{n,}	앞의 문자가 n개 이상 반복	(A B)	서브패턴과 OR 조건으로 A 또는 B

룰 옵션 : pcre 옵션

일반적인 pcre 옵션은 다음과 같습니다.

옵션	설명	비고
☆ i	대소문자를 구별하지 않음	nocase;
☆ s	PCRE의 특수문자인 점(.)은 줄바꿈 문자를 제외한 모든 문자를 나타내는데, 이 옵션을 사용하면 줄바꿈 문자까지 포함하여 패턴 매치	
☆ m	기본적으로 전체 문자열을 한 줄로 인식하는데, 이 옵션 사용 시 멀티라인으로 패턴 매치 가능	
x	탐지 문자열에서 모든 공백 문자를 무시함 단, 이스케이프 처리된 공백은 제외	
A	탐지 문자열을 패킷 페이로드의 첫번째 위치에서 패턴 매칭하며, PCRE의 특수문자인 캐럿(^)과 같은 기능	
E	마지막 문자열에서만 패턴 매칭하며 PCRE의 특수문자인 달러(\$)와 유사하지만, 이 옵션은 마지막 문자가 줄바꿈일 경우 매칭 안됨	
G	greedy하게 동작. 가능한 최대로 패턴 매칭	

룰 옵션 : pcre 옵션 (only snort)

SNORT에서만 사용할 수 있는 pcre 옵션은 다음과 같습니다.

옵션	설명	비고
☆ R	마지막에 패턴 매치된 문자열 이후부터 검색	distance:0;
☆ U	정규화 된 URI 버퍼에서 검색	http_uri;
B	정규화 되지 않은 원본 패킷에서 검색	rawbytes;
☆ P	HTTP 요청 메시지의 본문에서 검색	http_client_body;
☆ H	정규화 된 HTTP 요청 메시지 헤더에서 검색	http_header;
M	HTTP 메서드에서 검색	http_method;
☆ C	정규화 된 HTTP 요청과 응답 쿠키 헤더에서 검색	http_cookie
O	pcre match limit 값에 제한없이 탐지 수행 (default: 3500)	
☆ I	정규화 되지 않은 URI 버퍼에서 검색	http_raw_uri;
D	정규화 되지 않은 HTTP 요청 메시지 헤더에서 검색	http_raw_header;
K	정규화 되지 않은 HTTP 요청과 응답의 쿠키 헤더에서 검색	http_raw_cookie;
Y	HTTP 응답 상태 메시지에서 검색	http_stat_msg;
S	HTTP 응답 코드에서 검색	http_stat_code;

룰 옵션 : pcre 검증 도구

SNORT에서 pcre 문자열을 입력하기 전에 간단하게 검색할 수 있는 방법을 소개합니다.

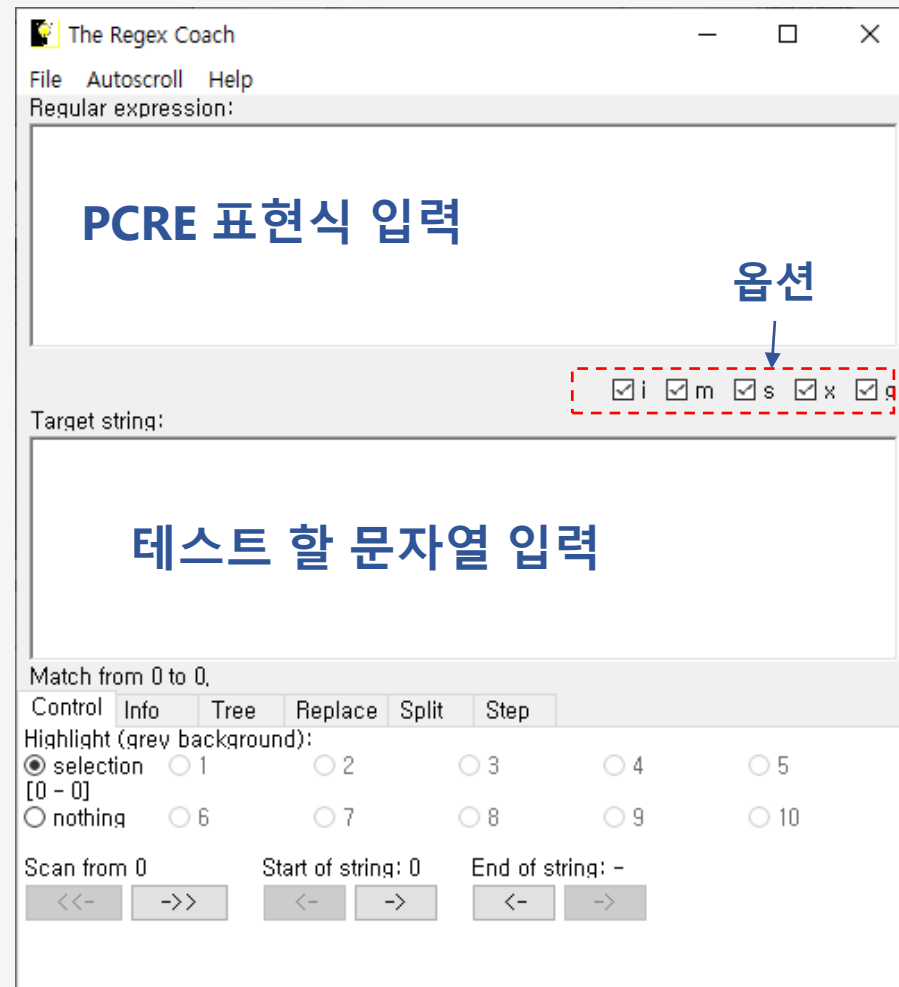
- 어플리케이션 (Freeware)

<http://weitz.de/files/regex-coach-0.8.5.tgz>

- 온라인 사이트

<https://regex101.com/>

<https://www.regextester.com/>



실습1 : pcre를 이용한 입력 값 검증

sample_sqli.pcap

[보기]의 룰에서 빈칸에 맞는 pcre 키워드를 넣어서 탐지 테스트를 해보세요.

[보기]

```
alert tcp any any -> any 80 (msg:"pcre keyword"; flow:to_server,established;  
content:"GET"; http_method; content:"/unlock_tasks.php"; nocase; http_uri;  
content:"cycle="; nocase; distance:0; http_uri; pcre:"  "; sid:1000200;)
```

[힌트] HTTP GET URI의 구조를 생각하고, PCRE의 부정형 옵션 **[^문자]** 와 ***** 를 이용해 보세요

GET **/admin/login.php?key=value&key=value** HTTP/1.1

실습2 : pcre를 이용한 Buffer Overflow 검증

sample_bof.pcap

[보기]의 룰에서 빈칸에 맞는 pcre 키워드를 넣어서 탐지 테스트를 해보세요.

[보기]

```
alert tcp any any -> any 80 (msg:"pcre keyword-2"; flow:to_server,established;  
content:"POST"; http_method; content:"/cgi/login.cgi"; nocase; http_uri;  
content:"name="; nocase; http_client_body; pcre:"  "; sid:1000201;)
```

[힌트] HTTP POST 요청의 구조를 생각하고, PCRE의 부정형 옵션 **[^문자]** 과 수량 한정자 **{n}** 를 이용해 보세요

POST **/admin/login.php** HTTP/1.1
Host: www.test.com

key=value&key=value

Part IV. 패킷 페이로드 검사하기 (byte_*)

룰 옵션 : byte_test

byte_test 키워드는 특정 위치의 값과 비교 연산이 필요하는데 사용합니다.

단순히 크기를 비교할 수도 있고, bit 단위로 and 와 or 연산 및 부정 연산자도 사용할 수 있습니다.

이 키워드는 어떤 취약점을 탐지하기 위해서 특정 위치의 값을 지정한 값(예: 0x7FFFFFFF)과 크기를 비교해서 해당 값이 지정한 값보다 큰 경우 공격으로 탐지하는 룰을 작성할 때 활용할 수 있습니다..

▪ 입력 형식

byte_test:<bytes to convert>, [!]<operator>, <value>, <offset> [, relative][, <endian>][, string, <number type>][, dce] [, bitmask <bitmask_value>];

룰 옵션 : byte_test

byte_test:<bytes to convert>, [!]<operator>, <value>, <offset> [, relative][, <endian>][, string, <number type>][, dce] [, bitmask <bitmask_value>];

옵션	입력 범위	설명
bytes to convert	1 ~ 10	패킷에서 비교할 대상의 바이트 수
operator	<, =, >, <=, >=, &, ^	크기 비교 및 비트 연산
value	0 ~ 4294967295	비교할 값
offset	-65535 ~ 65535	시작 위치로부터 건너뛰기 할 바이트 크기
relative	relative	이전에 탐지된 문자열의 끝으로 시작 위치 조정
endian	big, little	빅 엔디안(default) 또는 리틀 엔디안으로 처리
string	string	추출된 값을 문자열 형식으로 저장
number type	hex, dec, oct	추출된 문자열이 16진수(hex), 10진수(dec), 8진수(oct)로 표시
dce	dce	DCE/RPC 2 전처리기가 변환할 값의 바이트 순서 결정
bitmask_value	1 ~ 4 바이트 16진수 값	추출된 바이트에 AND 연산자를 적용 결과는 마스크의 후행 0 수와 같은 비트 수만큼 오른쪽으로 이동

byte_test 예제 (1/6)

[Rule]

```
alert tcp any any -> any any (msg:"byte_test Test"; content:" LSUB |22|"; byte_test:4,>,256,0,string,dec,relative;)
```

[Packet]



31 20 4C 53 55 42 20 22 22 20 7B 31 30 36 34 7D

1 LSUB "" {1064}

byte_test 예제 (2/6)

[Rule]

```
alert tcp any any -> any any (msg:"byte_test Test"; content:" LSUB |22|"; byte_test:4,>,256,0,string,dec,relative;)
```

[Packet]



31 20 4C 53 55 42 20 22 22 20 7B 31 30 36 34 7D

1 LSUB "" {1064}

byte_test 예제 (3/6)

[Rule]

```
alert tcp any any -> any any (msg:"byte_test Test"; content:" LSUB |22|"; byte_test:4,>,256,3,string,dec,relative;)
```

[Packet]



31 20 4C 53 55 42 20 22 22 20 7B 31 30 36 34 7D

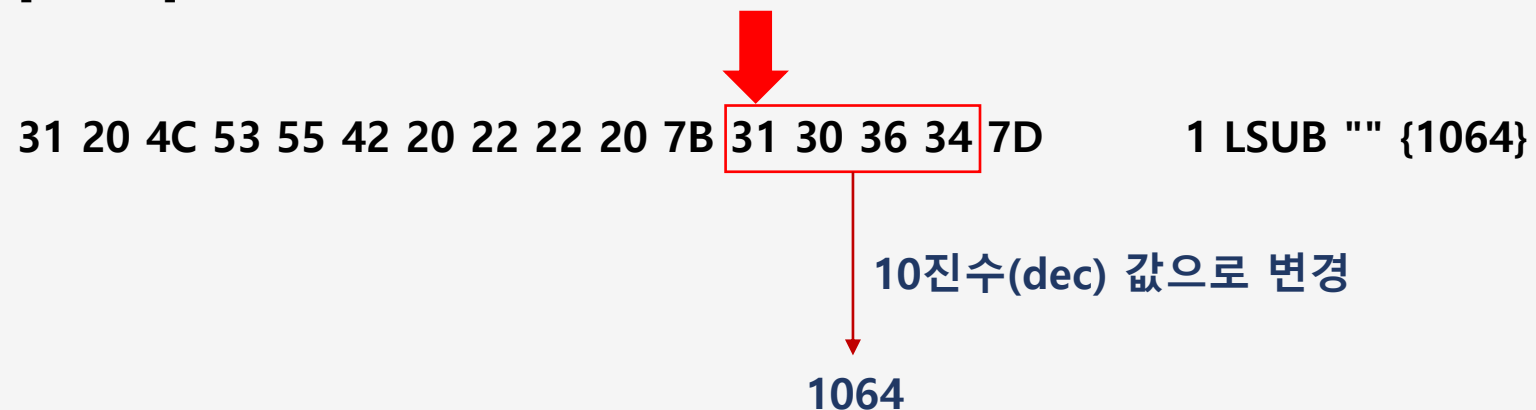
1 LSUB "" {1064}

byte_test 예제 (4/6)

[Rule]

```
alert tcp any any -> any any (msg:"byte_test Test"; content:" LSUB |22|"; byte_test:4,>,256,3,string,dec,relative;)
```

[Packet]



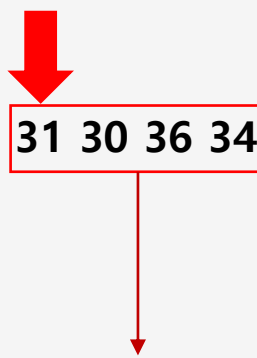
byte_test 예제 (5/6)

[Rule]

```
alert tcp any any -> any any (msg:"byte_test Test"; content:" LSUB |22|"; byte_test:4,>,256,0,string,dec,relative;)
```

[Packet]

31 20 4C 53 55 42 20 22 22 20 7B **31 30 36 34** 7D 1 LSUB "" {1064}



1064 > 256 결과가 'True' 이면 탐지

byte_test 예제 (6/6)

[Rule]

```
alert tcp any any -> any any (msg:"byte_test Test"; content:" LSUB |22|"; byte_test:4,>,256,0,string,dec,relative;)
```

[Packet]



31 20 4C 53 55 42 20 22 22 20 7B 31 30 36 34 7D

1 LSUB "" {1064}

실습 : byte_test를 이용한 값 비교

sample_post.pcap

[보기]의 룰에서 빈칸에 맞는 pcre 키워드를 넣어서 탐지 테스트를 해보세요.

[보기]

```
alert tcp any any -> any 80 (msg:"byte_test keyword"; content:"POST";  
http_method; content:"Content-Length:"; byte_test:  ; sid:1000300;)
```

룰 옵션 : byte_extract

byte_extract 키워드는 특정 위치의 값을 변수로 저장하면, 다른 키워드의 값으로 사용할 수 있습니다.

사용 가능한 키워드는 다음과 같습니다.

키워드	옵션
content	offset, depth, distance, within
byte_test	offset, value
byte_jump	offset
isdataat	offset

■ 입력 형식

byte_extract:<bytes_to_extract>, <offset>, <name> [, relative][, multiplier <multiplier value>][, <endian>][, string][, hex][, dec][, oct][, align <align value>][, dce][, bitmask <bitmask>];

byte_extract 예제 (1/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;  
byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]



0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00	MZ.....
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68Th
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f	is program canno
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20	t be run in DOS
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00	mode.....
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00PE.....
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

byte_extract 예제 (2/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;  
byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]

0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00	MZ.....
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68Th
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f	is program canno
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20	t be run in DOS
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00	mode.....
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00PE.....
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

byte_extract 예제 (3/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;  
byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]



0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

MZ.....
.....
.....
.....
.....Th
is program canno
t be run in DOS
mode.....
.....
.....
.....
.....
.....PE.....
.....

byte_extract 예제 (4/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;  
    byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]

0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00	MZ.....
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68Th
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f	is program canno
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20	t be run in DOS
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00	mode.....
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00PE.....
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

byte_extract 예제 (5/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;  
byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]

0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

MZ.....
.....
.....
.....
.....Th
is program cannot
be run in DOS
mode.....
.....
.....
.....
.....
.....PE.....
.....

little endian으로 변경
(f0 00 00 00 → 00 00 00 f0)

pe_offset = 0x000000f0

byte_extract 예제 (6/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;
byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]



0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00	MZ.....
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68Th
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f	is program canno
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20	t be run in DOS
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00	mode.....
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00PE.....
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00


pe_offset = 0x000000f0

byte_extract 예제 (7/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;  
    byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]

0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00	MZ.....
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68Th
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f	is program canno
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20	t be run in DOS
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00	mode.....
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00PE.....
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

pe_offset = 0x000000f0

byte_extract 예제 (8/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;
    byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]

0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00	MZ.....
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68Th
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f	is program canno
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20	t be run in DOS
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00	mode.....
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f3 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00PE.....
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

pe_offset = 0x000000f0

byte_extract 예제 (9/9)

[Rule]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ"; depth:2;
byte_extract:4,60,pe_offset,little; content:"PE|00 00|"; offset:pe_offset; depth:4;)
```

[Packet]

0000	4d 5a 90 00 03 00 00 00	04 00 00 00 ff ff 00 00	MZ.....
0010	b8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
0020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00	00 00 00 00 f0 00 00 00
0040	0e 1f ba 0e 00 b4 09 cd	21 b8 01 4c cd 21 54 68Th
0050	69 73 20 70 72 6f 67 72	61 6d 20 63 61 6e 6e 6f	is program canno
0060	74 20 62 65 20 72 75 6e	20 69 6e 20 44 4f 53 20	t be run in DOS
0070	6d 6f 64 65 2e 0d 0d 0a	24 00 00 00 00 00 00 00	mode.....
0080	b6 b5 e8 7c f2 d4 86 2f	f2 d4 86 2f f2 d4 86 2f
0090	fb ac 05 2f fd d4 86 2f	fb ac 15 2f d1 d4 86 2f
00a0	f2 d4 87 2f 03 d6 86 2f	9d a2 18 2f d9 d4 86 2f
00b0	9d a2 2c 2f 41 d4 86 2f	9d a2 2d 2f 85 d5 86 2f
00c0	9d a2 29 2f f1 d4 86 2f	9d a2 1c 2f f3 d4 86 2f
00d0	9d a2 1b 2f f2 d4 86 2f	52 69 63 68 f2 d4 86 2f
00e0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00PE.....
00f0	50 45 00 00 4c 01 05 00	8d da 1a 54 00 00 00 00

pe_offset = 0x000000f0

실습 : byte_extract를 이용한 변수 값 추출과 활용

sample_exe_down.pcap

[보기]의 룰에서 빈칸에 맞는 pcre 키워드를 넣어서 탐지 테스트를 해보세요.

[보기]

```
alert tcp any any -> any any (msg:"byte_extract Test"; file_data; content:"MZ";  
depth:2; byte_extract: ; content:"PE|00 00|"; offset:pe_offset; depth:4;  
sid:1000301;)
```

[힌트] 실행파일(PE)의 구조를 확인해 보세요.

[별첨3. 실행파일의 구조](#)

Part V. 임계치 기반 탐지

룰 옵션 : threshold

threshold 키워드는 네트워크 트래픽에서 발생하는 이벤트의 수를 제한하는 기능입니다.

이 기능은 특정 시간 동안 로그에 기록되는 특정 이벤트의 수를 제한하므로, 노이즈가 많은 규칙에 대한 로그 경고 수를 줄일 수 있고, Bruteforce 또는 DoS 공격을 탐지하는데 활용할 수 있습니다.

■ 입력 형식

threshold: type <limit|threshold|both>, track <by_src|by_dst>, count <c>, seconds <s>;

키워드	옵션	설명
type	• threshold	지정한 시간(seconds) 동안 설정한 count가 될 때마다 이벤트 한 번 발생
	• limit	지정한 시간(seconds) 동안 동일한 이벤트가 발생하면 설정한 개수만큼만 출력 후 종료
	• both	지정한 시간(seconds) 안에 동일한 이벤트가 설정한 개수가 되었을 때, 이벤트 한 번 발생
track	• by_src	출발지 IP 기준으로 count
	• by_dst	목적지 IP 기준으로 count
count		제한 시간 동안 발생하는 시그니처 매칭 횟수
seconds		카운트가 누적되는 시간(초)

룰 옵션 : threshold

sample_threshold.pcap

실습을 통해서 이벤트가 몇 번 발생하는지 관찰해 보세요.

1 threshold 미사용

```
alert icmp any any -> any any (msg:"ICMP Detection"; itype:8; icode:0; sid:1100100;)
```

2 threshold 사용

```
alert icmp any any -> any any (msg:"threshold Test"; itype:8; icode:0; threshold:type threshold, track by_src, count 5, seconds 60; sid:1100101;)
```

3 limit 사용

```
alert icmp any any -> any any (msg:"limit Test"; itype:8; icode:0; threshold:type limit, track by_src, count 3, seconds 60; sid:1100102;)
```

4 both 사용

```
alert icmp any any -> any any (msg:"both Test"; itype:8; icode:0; threshold:type both, track by_src, count 5, seconds 60; sid:1100103;)
```

룰 옵션 : detection_filter

threshold 키워드는 네트워크 트래픽에서 발생하는 이벤트의 수를 제한하는 기능입니다.

이 기능은 특정 시간 동안 로그에 기록되는 특정 이벤트의 수를 제한하므로, 노이즈가 많은 규칙에 대한 로그 경고 수를 줄일 수 있고, Bruteforce 또는 DoS 공격을 탐지하는데 활용할 수 있습니다.

■ 입력 형식

detection_filter: track <by_src|by_dst>, count <c>, seconds <s>;

키워드	옵션	설명
track	<ul style="list-style-type: none">• by_src• by_dst	출발지 IP 기준으로 count 목적지 IP 기준으로 count
count		제한 시간 동안 발생하는 시그니처 매칭 횟수
seconds		카운트가 누적되는 시간(초)

룰 옵션 : detection_filter

sample_threshold.pcap

실습을 통해서 이벤트가 몇 번 발생하는지 관찰해 보세요.

1 detection_filter 미사용

```
alert icmp any any -> any any (msg:"ICMP Detection"; itype:8; icode:0; sid:1100200;)
```

2 detection_filter 사용

```
alert icmp any any -> any any (msg:"detection_filter Test"; itype:8; icode:0; detection_filter:track by_src, count 3, seconds 60; sid:1100201;)
```

C:\Snort\etc 경로의 threshold.conf를 설정하면, detection_filter에서 발생한 이벤트를 기준으로 threshold 키워드와 동일한 결과를 얻을 수 있습니다. 단, sid 당 1개만 설정 가능합니다.

```
event_filter gen_id 1, sig_id 1100201, type threshold, track by_src, count 1, seconds 60
# event_filter gen_id 1, sig_id 1100201, type limit, track by_src, count 1, seconds 60
# event_filter gen_id 1, sig_id 1100201, type both, track by_src, count 1, seconds 60
```

정보보안기사 문제 살펴보기

정보보안기사 3회 기출문제

[문제] “/administrator”라는 문자열이 포함되어 있는 경우 “Web Scan Detected”란 메시지 로깅을 위한 Snort Rule은 각각 무엇인가?

```
alert tcp any any -> 192.168.0.1 ( A ) ( ( B ) : “/administrator”; ( C ) : “Web Scan Detected” ; )
```

[정답]

정보보안기사 7회 기출문제

[문제] Snort 정책에서 10바이트에서 12바이트 중 00FF 바이트에 해당하는 내용을 찾으려고 한다. 보기의 rule에 빈칸을 채워 rule을 완성하시오.

```
alert tcp any any -> any any (( A ):"|00FF|"; ( B ):9; ( C ):2;)
```

[정답]

정보보안기사 11회 기출문제

[문제] 모든 네트워크 대역에서 Telnet으로 접속하는 패킷 중 14번째 자리까지 'anonymous'가 포함된 트래픽에 대해서 'Dangerous' 메시지로 경고하는 snort rule을 만드시오.

[정답]

정보보안기사 12회 기출문제

[문제] Snort의 각 규칙은 고정된 헤더와 옵션을 가지고 있다.

패킷의 payload 데이터를 검사할 때 사용되는 옵션에 포함되지 않는 필드는?

- ① ttl
- ② content
- ③ depth
- ④ offset

[정답]

정보보안기사 13회 기출문제

[문제] Snort Rule 설정 의미를 설명하시오.

```
alert tcp any any <> any ①[443,465,523] (②content:"|18 03 00|"; depth: 3;  
③content:"|01|"; distance: 2; within: 1; ④content:!"|00|"; within: 1;  
⑤msg: "SSLv3 Malicious Heartbleed Request V2"; sid:1;)
```

정답

정보보안기사 14회 기출문제

[문제] XSS 공격을 탐지하기 위한 Snort rule에 대하여 다음 물음에 답하시오.

```
alert any any -> any 80 (msg:"XSS"; content:"GET"; offset:1; depth:3;  
content:"/Login.php<script>XSS"; distance:1; )
```

(1) content : "GET", offset : 1, depth : 3 의미

(2) content:"/Login.php XSS"; distance 1; 의미

(3) 위의 룰이 탐지되지 않는다면, 어떻게 수정해야 하는지 기술

정보보안기사 22회 기출문제

[문제] Snort에서는 대량의 패킷에 대응하기 위하여 Threshold 옵션을 type(action 수행 유형), track(소스/목적지 IP), count(횟수), second(시간)으로 설정할 수 있다.
이 중 threshold type 3가지를 기술하시오.

```
threshold <(1) | (2) | (3)>, track <by_src | by_dst>, count <c>, seconds <s>
```

정답

Snort Rule

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"SERVER-WEBAPP mojoPortal Forums txtTitle cross site scripting attempt";  
flow:to_server,established; content:"/Forums/EditForum.aspx"; fast_pattern:only; http_uri; content:"txtTitle="; http_client_body;  
pcre:"/(%24|%(25)?24)txtTitle=[^&]*?([%22%27%3c%3e%28%29]|%(25)?(22|27|3c|3e|28|29)|script|onload|src)/Pi"; metadata:policy max-  
detect-ips drop, policy security-ips drop, service http; reference:url,www.exploit-db.com/exploits/49184; classtype:attempted-user; sid:61082;  
rev:1;)
```

Snort Rule

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"SERVER-WEBAPP mojoPortal Forums txtTitle cross site scripting attempt";
flow:to_server,established;

content:"/Forums/EditForum.aspx"; fast_pattern:only; http_uri;
content:"txtTitle="; http_client_body;
pcr:"/(Wx24|%(25)?24)txtTitle=[^&]*?([Wx22Wx27Wx3cWx3eWx28Wx29]|%(25)?(22|27|3c|3e|28|29)|script|onload|src)/Pi";

metadata:policy max-detect-ips drop, policy security-ips drop, service http;
reference:url,www.exploit-db.com/exploits/49184;
classtype:attempted-user;
sid:61082; rev:1;)
```

별첨1. Pre-authenticated SQL injection in GLPI <= 9.3.3

Presentation of GLPI

"GLPI ITSM is a software for business powered by open source technologies. Take control over your IT infrastructure: assets inventory, tickets, MDM."

The issue

Synacktiv discovered that GLPI exposes a script (`/scripts/unlock_tasks.php`) that not correctly sanitize usercontrolled data before using it in SQL queries. Thus, an attacker could abuse the affected feature to alter the semantic original SQL query and retrieve database records.

Source Code : unlock_tasks.php

```
if (isset($_GET['cycle'])) {  
    $cycle = $_GET['cycle'];  
} else {  
    $cycle = 25;  
}  
[...중략...]  
$crontask = new Crontask();  
$query = "SELECT `id`, `name`  
FROM `glpi_crontasks`  
WHERE `state` = '".Crontask::STATE_RUNNING."  
AND unix_timestamp(`lastrun`) + $cycle * `frequency` <  
unix_timestamp(now())";
```



별첨2. URI 정규화

정규화된 URI란, URI를 일관된 방식으로 수정하고 표준화하는 과정을 말합니다.

꼭 알아야 할 정규화 내용만 소개해 드리고, 상세한 내용은 SNORT 공식 홈페이지에서 HttpInspect 문서를 참고하세요.

정규화	설명	예시
% decoding	'%XX' 문자 디코딩	'%41' → 'A' 로 치환
%u decoding	'%uXXXX' 문자 디코딩	'%u0041' → 'A' 로 치환
Multiple slash	다중 '/' 경로 디코딩	'////////' → '/' 로 치환
Double decoding	이중 디코딩	'%2541' → '%41' → 'A'

* HttpInspect 문서 : https://www.snort.org/document/readme-http_inspect



별첨3. 실행 파일의 구조

실행파일을 탐지하기 위해서는 매직 넘버 값('MZ')과 PE 헤더를 찾아야 합니다.

PE 헤더의 위치는 가변적이어서, 이 헤더를 찾기 위해서는 PE 헤더의 위치 정보를 가지고 있는 e_lfanew 값을 찾아야 합니다.

```
#define IMAGE_DOS_SIGNATURE          0x4D5A      // MZ

typedef struct IMAGE_DOS_HEADER {          // DOS .EXE header
    WORD e_magic;                          // Magic number
    WORD e_cblp;                          // Bytes on last page of file
    WORD e_cp;                            // Pages in file
    WORD e_crlc;                          // Relocations
    WORD e_cparhdr;                       // Size of header in paragraphs
    WORD e_minalloc;                      // Minimum extra paragraphs needed
    WORD e_maxalloc;                      // Maximum extra paragraphs needed
    WORD e_ss;                            // Initial (relative) SS value
    WORD e_sp;                            // Initial SP value
    WORD e_csum;                          // Checksum
    WORD e_ip;                            // Initial IP value
    WORD e_cs;                            // Initial (relative) CS value
    WORD e_lfarlc;                        // File address of relocation table
    WORD e_ovno;                          // Overlay number
    WORD e_res[4];                        // Reserved words
    WORD e_oemid;                         // OEM identifier (for e_oeminfo)
    WORD e_oeminfo;                       // OEM information; e_oemid specific
    WORD e_res2[10];                      // Reserved words
    LONG e_lfanew;                        // File address of new exe header
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

File Disk Edit **e_magic** ns Registry Bookmarks Misc Help

e_lfanew

0000	4d 5a 90 00	03 00 00 00	04 00 00 00	ff ff 00 00	MZ.....yy..
0010	b8 00 00 00	00 00 00 00	40 00 00 00	00 00 00 00@.....
0020	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 000...
0030	00 00 00 00	00 00 00 00	00 00 00 00	f8 00 00 00	..°.°.f!.LI!Th
0040	0e 1f ba 0e	00 b4 09 cd	21 b8 01 4c	cd 21 54 68	is program canno
0050	69 73 20 70	72 6f 67 72	61 6d 20 63	61 6e 6e 6f	t be run in DOS
0060	74 20 62 65	20 72 75 6e	20 69 6e 20	44 4f 53 20	mode....\$......
0070	6d 6f 64 65	2e 0d 0d 0a	24 00 00 00	00 00 00 00	2.T.vc:ävç:ävç:ä
0080	32 02 54 b7	76 63 3a e4	76 63 3a e4	76 63 3a e4	..@äpc:ä..9âtç:ä
0090	7f 1b a9 e4	70 63 3a e4	19 07 39 e5	74 63 3a e4	vc;äQc:ä..;ä.c:ä
00a0	76 63 3b e4	51 63 3a e4	19 07 3b e5	7f 63 3a e4	..>âtç:ä..3âtç:ä
00b0	19 07 3e e5	65 63 3a e4	19 07 33 e5	74 63 3a e4	..?âtç:ä..Ääwc:ä
00c0	19 07 3f e5	74 63 3a e4	19 07 c5 e4	77 63 3a e4	..8äwc:äRichvc:ä
00d0	19 07 38 e5	77 63 3a e4	52 69 63 68	76 63 3a e4
00e0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00PE..L...
00f0	00 00 00 00	00 00 00 00	50 45 00 00	4c 01 05 00	..°.....à...
0100	60 92 da 8c	00 00 00 00	00 00 00 00	e0 00 02 01T.....
0110	0b 01 0e 0c	00 10 00 00	00 54 00 00	00 00 00 00@.
0120	60 1b 00 00	00 10 00 00	00 20 00 00	40 00 00 00
0130	00 10 00 00	00 02 00 00	0a 00 00 00	0a 00 00 00

PE Header