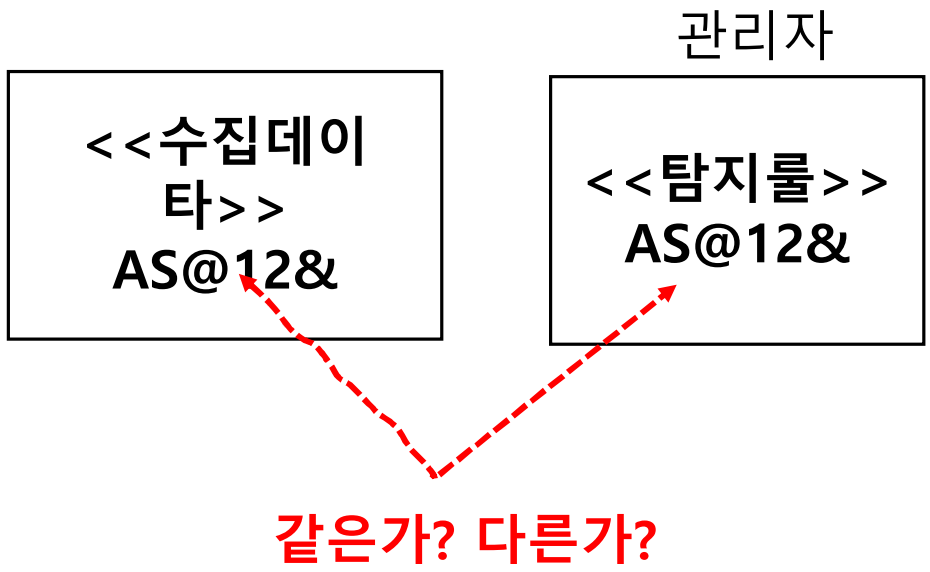
# Snort Rule

# Snort(스노트)




- 1998년 마틴로쉬에 의해 개발
- 오픈소스로 시그니처 기반 NIDS
  - 네트워크 패킷을 수집하여 트래픽을 모니터링
  - 준비된 규칙과 비교하여 침입탐지 및 경로를 발생

  * **시그니처(signature) 기반이란 : 침입탐지를 문자열로 판단하는것**

  **(패킷 데이터에서 악의적인 문자열을 탐지하여 침입여부를 결정)**

- 오늘날 침입탐지시스템의 대명사로 사용

# Suricata(수리카타)

- 2010년 OISF 단체에서 오픈 소스 프로젝트로 개발한 NIDS/IPS
- Snort의 단점을 개선하고 장점을 수용
  - 멀티 코어 및 멀티 스레드 지원 : 대용량 트래픽 실시간 처리(성능향상)
  - Snort Rule 완변 호환 및 대부분의 기능 지원
  - 하드웨어 벤더의 개발 지원으로 하드웨어 가속 지원
  - 스크립트 언어(Lua) 지원

# Snort

- 오픈 소스로 시그니처 기반 네트워크 침입탐지 시스템 (NIDS)
- 네트워크 패킷을 수집하여 트래픽을 모니터링하고 준비된 규칙과 비교하여

  침입 탐지 및 경고를 발생
- **시그니처 기반이란 침입탐지를 문자열로 판단한다는 의미**

  **- 악의적인 문자열을 탐색하여 침입여부를 결정**


- 스니퍼 : 네트워크 패킷을 수집
- 패킷 디코더 : 수집된 패킷은 디코더로 전처리기와 탐지 엔진이 파싱할 수 있도록 정규화
- 전처리기 : 특정 행위가 발결된 패키을 탐색 엔진에 전송
- 탐색엔진 : 해당 패킷이 스노트 규칙에 매칭 되는 지 확인
- 경고/로그 : 스노트 규칙에 매칭 된다면 콘솔 창이라 분석 도구에 경고를 출력하고 기록

# Snort Rule

- Intruder Detection Rule
- NIDS나 IPS는 Pattern으로 정의된 rule을 기반으로 탐지한다.

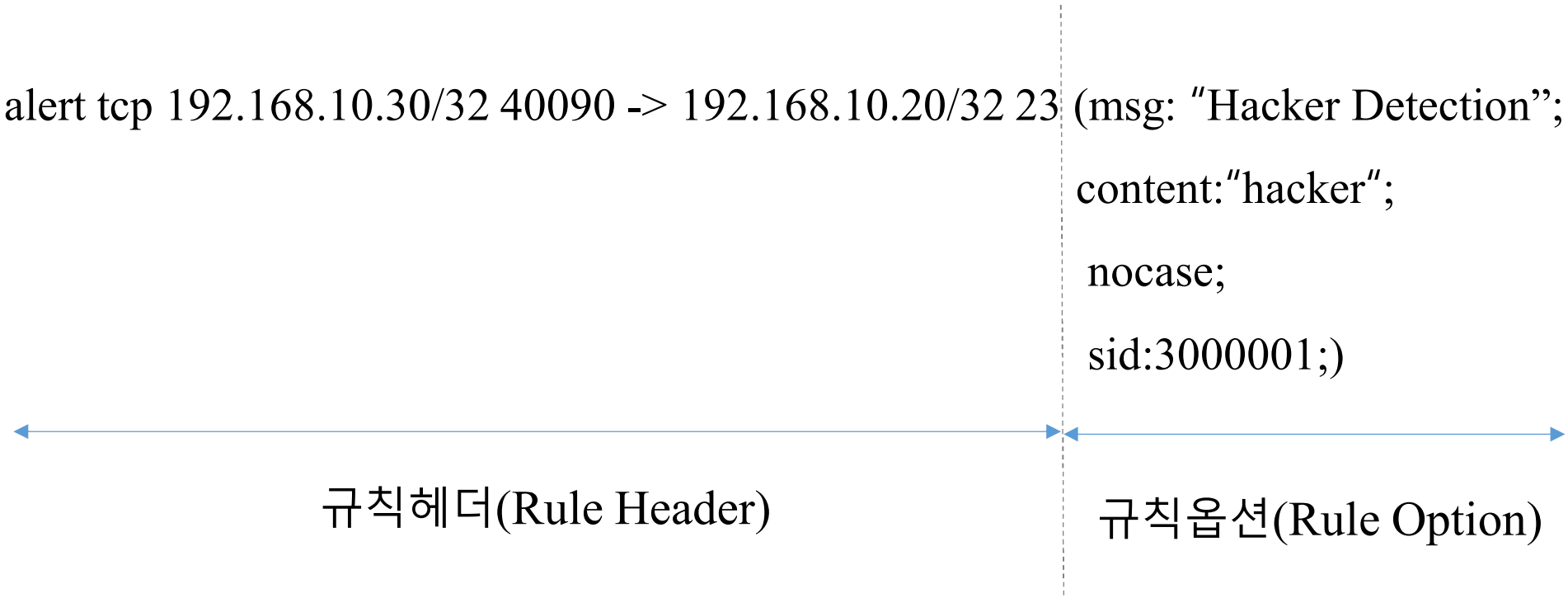
  - Packet의 Payload를 검사하는 방식으로 공격을 탐지한다.

  - 기본적으로 signature(혹은 pattern)을 비교 혹은 검색하는 방식으로 탐지한다.

  - 정규표현식(Regular expression)으로 탐지 규칙을 규정할 수 있다.

  - DDoS 공격은 단위 시간 동안의 '발생량'을 기반으로 탐지한다.

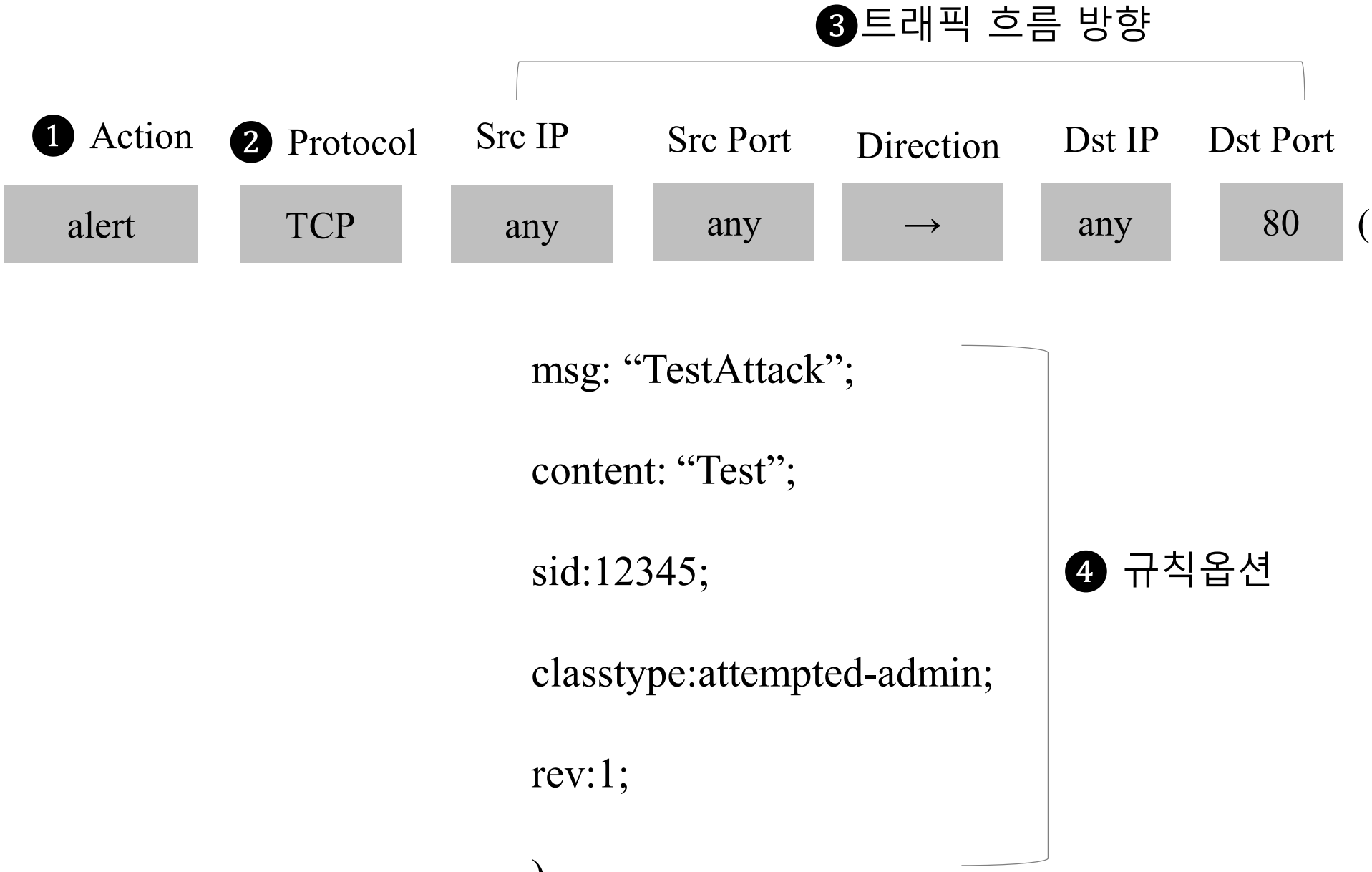
# Snort 시그니처 기반의 IDS 의 Detection Rule 기본 구조



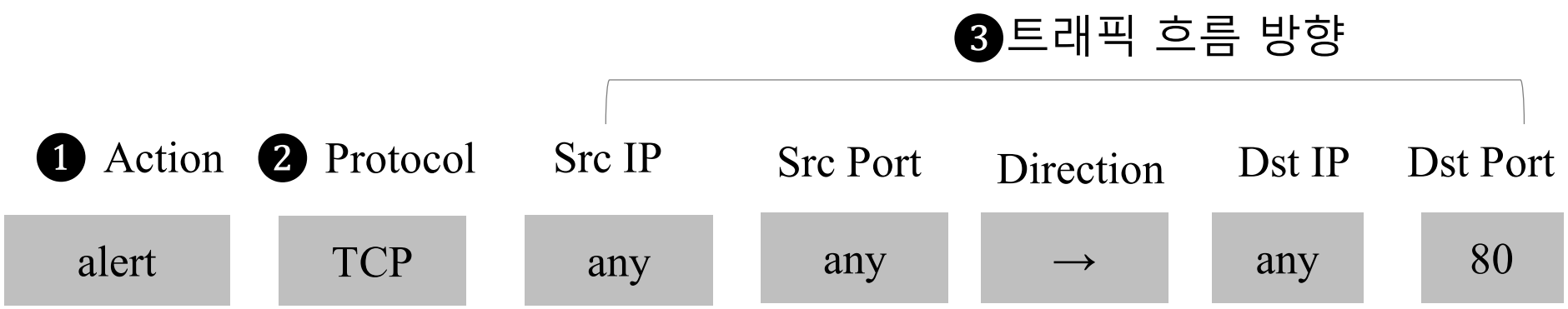
# Rule sample

alert tcp any → 192.168.1.0/24 111 ( msg: “mounted access”; content: “|00 01 86 a5|”)

| Name | Descriptions |
|---|---|
| Action | Alert |
| Protocol | TCP |
| Source | 룰 적용 대상 출발지(공격자)IP 주소 및 포트는 '전체 ' |
| Direction | 특정 네트워크  Inbound |
| Destination | 192.168.1.x 네트워크 111 포트에 대한 접근 |
| Message | “mounted access” |
| Pattern | TCP payload에서 Hexa 스트링 0x00, 0x01, 0x86, 0xA5 패턴을 찾는다. |


❸트래픽 흐름 방향
❶ Action
❷ Protocol
Src IP
Src Port
Direction
Dst IP
Dst Port
alert
TCP
any
any
→
any
80
(
msg: “TestAttack”;
content: “Test”;
sid:12345;
classtype:attempted-admin;
rev:1;
)
❹ 규칙옵션

# Snort Rule Header



# ❶ Snort Rule Header – Action

| Option | Role |
|---|---|
| alert | 경고를 발생한다. |
| log | 패킷을 로그로 저장한다. |
| pass | 패킷을 무시한다. |
| active | 경고를 발생시킨 다음 다른 동적 규칙을 활성화 한다. |
| dynamic | Active 옵션으로 활성화 된다. |

*In-line 모드로 IDS가 배치된 경우

| Action | Description |
|---|---|
| drop | 패킷을 차단 한 후 로그로 저장한다. |
| reject | TCP의 경우, 차단 및 로그 저장 후 세션을 리셋(RST 전송)한다.<br>UDP의 경우 차단 및 로그 저장 후 ICMP port unreachable 메시지를 전송한다. |
| sdrop | 패킷을 차단하지만 로그는 남기지 않는다. |

# ❷ Snort Rule Header – Protocol

| Option | Role |
|---|---|
| tcp | TCP 프로토콜에 적용 |
| udp | UDP 프로토콜에 적용 |
| icmp | ICMP 프로토콜에 적용 |
| ip | IP 프로토콜에 적용 |

# ❸ Snort Rule Header – IP, Port

| Option | Role | |
|---|---|---|
| IP | any | 모든 IP 주소 |
| | 1.1.1.1 | 특정 IP 주소 |
| | [1.1.1.1, 2.2.2.2] | 여러 IP 주소 |
| | [1.1.1.1/24] | 특정 IP 주소 |
| PORT | Any | 모든 포트 번호 |
| | 80 | 특정 포트 번호 |
| | 1:1024 | 1~1024 번 포트 범위 |
| | 80: | 80 번 이상 범위 |
| | :1024 | 1024번 이하 범위 |
| | !80 | 80번을 뺀 나머지 |
| → | 단반향 | |
| <> | 양방향 | |

## ❹ 규칙 옵션

- 규칙 헤더에 해당하는 패킷 중 특정 패턴(문자열)을 정의해 놓은 영역
- 옵션 종류
  - 일반옵션
  - 흐름 옵션
  - 페이로드
  - HTTP 관련 옵션 등
- 옵션들은 ';'(세미콜론)으로 구분

# 일반 옵션

| Action | Protocol | Src IP | Src Port | Direction | Dst IP | Dst Port | |
|---|---|---|---|---|---|---|---|
| alert | TCP | any | any | → | any | 80 | ( |

msg: “TestAttack”;

sid:12345;

classtype:attempted-admin;

rev:1;

)

❹ 일반 옵션

# 일반 옵션

- 규칙에 대한 정보를 제공하는 옵션
- 검색하는 동안 어떤 영향도 미치지 않음

| | |
|---|---|
| **msg** | • 규칙이 탐지될 경우 출력되는 메시지<br>• 공격유형과 정보를 기록 |
| **sid** | • 규칙 식별자로 모든 규칙은 반드시 식별 번호를 가짐<br>• 예약된 식별자 : 0~2,999,999<br>• Local.rules에는 3,000,000이상부터 사용 |
| **rev** | • 규칙의 수정 버전을 나타냄<br>• 규칙이 수정 시 1 씩 증가 |
| **classtype** | • 규칙을 분류하는 옵션<br>• 클래스 명은 classfication.config 파일에 정의 |
| **priority** | • 규칙의 우선순위 지정<br>• 1 ~10까지의 수 사용, 숫자가 작을수록 높은 우선순위를 가짐 |

# Payload 옵션

- 악성 패킷을 탐지하는 옵션

| | |
|---|---|
| content | 매칭할 문자열 지정 |
| pcre | 문자열로 표현하기 어려운 것들을 정규 표현식을 이용하여 정의 할 경우 사용 |

- 문자열 지정 **content: "administrator";**
- 숫자 지정 **content: "|121212|";**
- 정규표현식 지정 **pcre: "/^select/";**

"/^select/"; 검색할 문자열 중 가장 앞에 위치한 select 문자가 있는 경우 매치

# Payload 옵션

| | |
|---|---|
| **content** | 매칭할 문자열 지정 |
| nocase | 대소문자 구별하지 않고 매칭 |
| offset | 매칭할 문자열의 위치 지정 |
| depth | 문자열의 범위 지정 |
| distance | Content 옵션값 이후 탐색할 위치 지정 |
| within | Content옵션값 이후의 탐색할 범위를 지정 |
| **pcre** | 문자열로 표현하기 어려운 것들을 정규 표현식을 이용하여 정의 할 경우 사용 |

| Option | Role | Example |
| --- | --- | --- |
| nocase | 대/소문자 구별하지 않음 | |
| offset | 패킷의 데이터 영역이 시작되는 지점을 기준으로 검사 시작 위치 지정 | (content: “a”; offset:1) |
| depth | Offset으로 시작된 검사의 종료 위치 지정(검사 종료 절대 위치) | (content: “a”; offset:1 ; depth:1) |

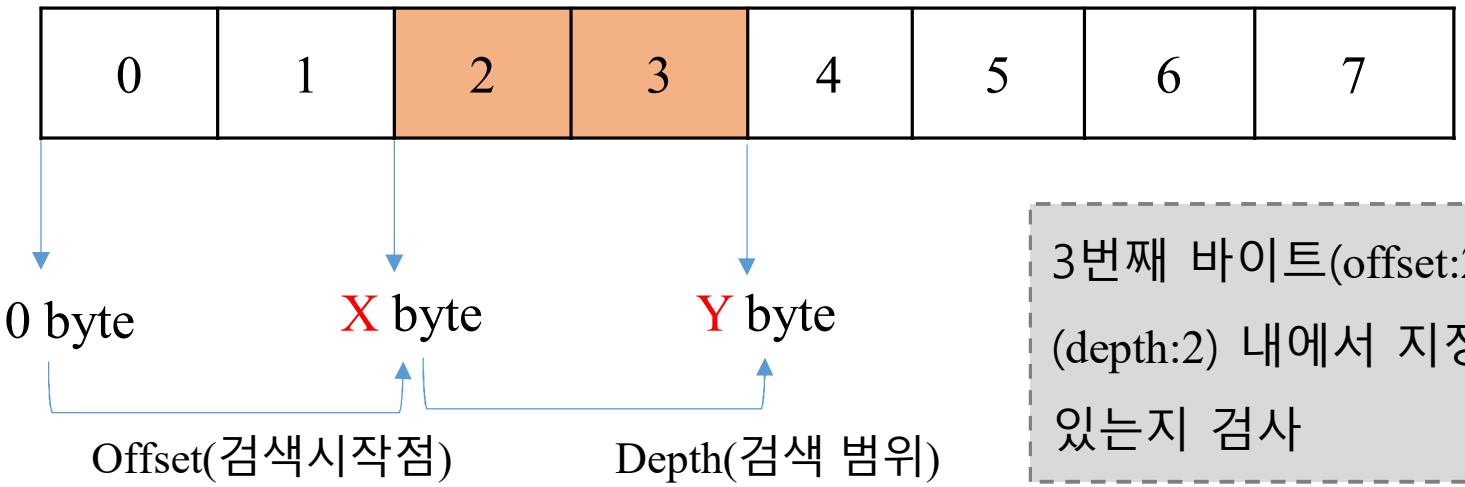
- **offset**

  - content 패턴을 검사 할 시작 위치

  - 첫 번째 바이트 위치가 0부터 시작

- **depth**

  - offset부터 몇 바이트까지 검사할 것인지 지정




3번째 바이트(offset:2)부터 2바이트 범위(depth:2) 내에서 지정된 문자열 패턴이 있는지 검사

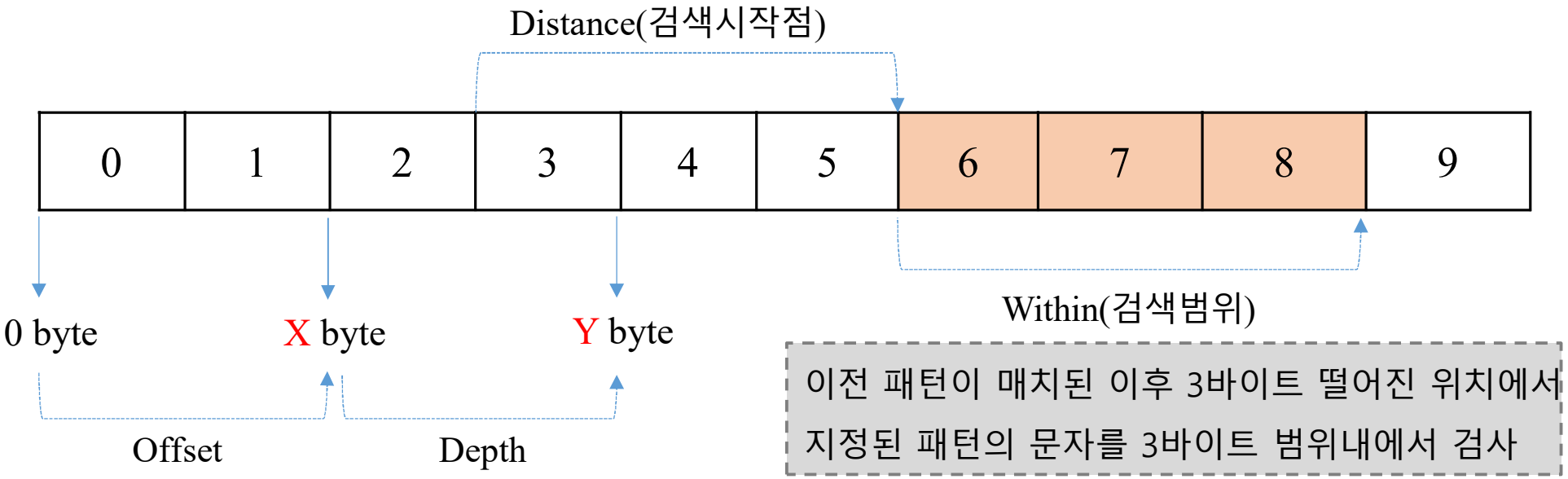
| Option | Role | Example |
| --- | --- | --- |
| distance | 이전 패턴 검사가 종료된 시점을 기준으로 검사 시작 위치 지정 | (content:“a”; content:“b”; distance:1;) |
| within | Distance로 시작된 검사의 종료 위치 지정(검사 종료 상대 위치) | (content:“a”; content:“b”; distance:1; within:1;) |
| rawbytes | 인코딩 된 문자열 패턴의 디코딩 전처리기와 관계없이 Hexa 코드로 검사 | (content: \|3B 20\|; rawbytes; ) |

- distance

  - 이전 content 패턴이 매치된 경우, 매치된 바이트로부터 몇 바이트 떨어진 위치에서 다음 content를 검사 할 것인지 지정

- within

  - distance부터 몇 바이트 범위 내에서 지정된 패턴을 검사할 것인지 지정



| Option | Role | Example |
| --- | --- | --- |
| http_client_body | 패턴 검사 범위를 웹요청(POST) 메시지 본문(body)으로 제한 | (content: “a”; http_client_body;) |
| http_cookie | 검사 범위를 cookie로 제한 | (content: “a”; http_cookie;) |
| http_header | 검사 범위를 HTTP 헤더 영역으로 제한 | (content: “Host”; http_header;) |
| http_method | 검사 범위를 웹 요청 메소드로 제한 | (content: “GET”; http_method;) |
| http_uri | 검사 범위를 웹 요청 URI로 제한 | (content: “a”; http_uri;) |

| Option | Role | Example |
| --- | --- | --- |
| http_state_code | 검사 범위를 웹 응답 코드 번호 영역으로 제한 | (content: “200”; http_state_code;) |
| http_stat_msg | 검사범위를 웹 응답 코드 메시지 영역으로 제한 | (content: “OK”; http_state_msg;) |
| fast_pattern | Contents 옵션이 두 개 이상 사용 될 때 검색 우선 순위를 조정<br>fast_pattern:only<br>중복 검사 방지<br>fast_pattern:offse값<br>우선 검사할 문자열의 검색위치<br>fast_pattern:length값<br>우선 검사할 문자열의 검색 범위 | (content: “aa”; content: “b”;<br>fast_pattern;)<br>aa에서 b 문자열을 먼저 검사 |

| Option | Role | Example |
|---|---|---|
| i | Content의 nocase와 동일 | (content: “a”; pcre: “/(B\|C)/i”;) |
| S | 메타문자'.'과 달리 공백문자까지 포함 | (content: “a”; pcre: “/\b./s”;) |
| m | 줄 바꿈 문자를 무시하고 여러 행을 한 행으로 이어진 문자열로 처리 | (content: “a”; pcre: “/^bc/m”;) |
| R | Content의 distance와 동일 | |
| B | Content의 rawbytes와 동일 | (content: “a”; pcre: “/\x3B\x20/B”;) |

# Header Rule Option

| Option | Role | Example |
|---|---|---|
| fragbits | 단편화 여부 검사<br>M(More fragment)<br>D(Don’t fragment),<br>R(Reserved bit) | fragbits:M; |
| fragoffset | 단편화된 패킷의 위치 검사 | fragbits:M gragoffset:0; |
| ttl | ttl값 검사 | ttl:=128 |
| tos | TOS값 검사 | tos:4 |
| id | IP 헤더 ID 값 검사 | id:12345; |

# Header Rule Option

| Option | Role | Example |
|---|---|---|
| Ipopts | IP헤더 옵션 값 검사 | |
| dsize | 패킷 데이터 영역 길이(byte) 검사 | dsize:<1024 |
| flow | TCP stream 전 처리로 패킷 방향 정의 | flow:from_client;<br>from_client(to_server),<br>from_server(to_client)<br>established |
| seq | 순서번호 | |
| ack | 응답값 | |
| window | TCP 헤더 윈도우값 | window:55555 or window:!33333 |
| sameip | 출발지 목적지가 동일한 IP인지 조사 | |

# Threshold

| 로그 발생 타입 | 로그 발생 기준 | 로그 발생 예시 |
|---|---|---|
| threshold:type **threshold**,<br>count 100, seconds 2; | 패킷양 | 2초내에 패킷 100개 : 로그 1개<br>2초내에 패킷 200개 : 로그 2개<br>4초내에 패킷 400개 : 로그 4개 |
| threshold:type **limit**,<br>count 100, seconds 2; | 임계시간 | 2초내에 패킷 100개 : 로그 1개<br>2초내에 패킷 200개 : 로그 1개<br>4초내에 패킷 400개 : 로그 2개 |
| threshold:type **both**,<br>count 100, seconds 2; | IP | 2초내에 패킷 100개 : 로그 1개<br>2초내에 패킷 200개 : 로그 1개<br>4초내에 패킷 400개 : 로그 1개 |

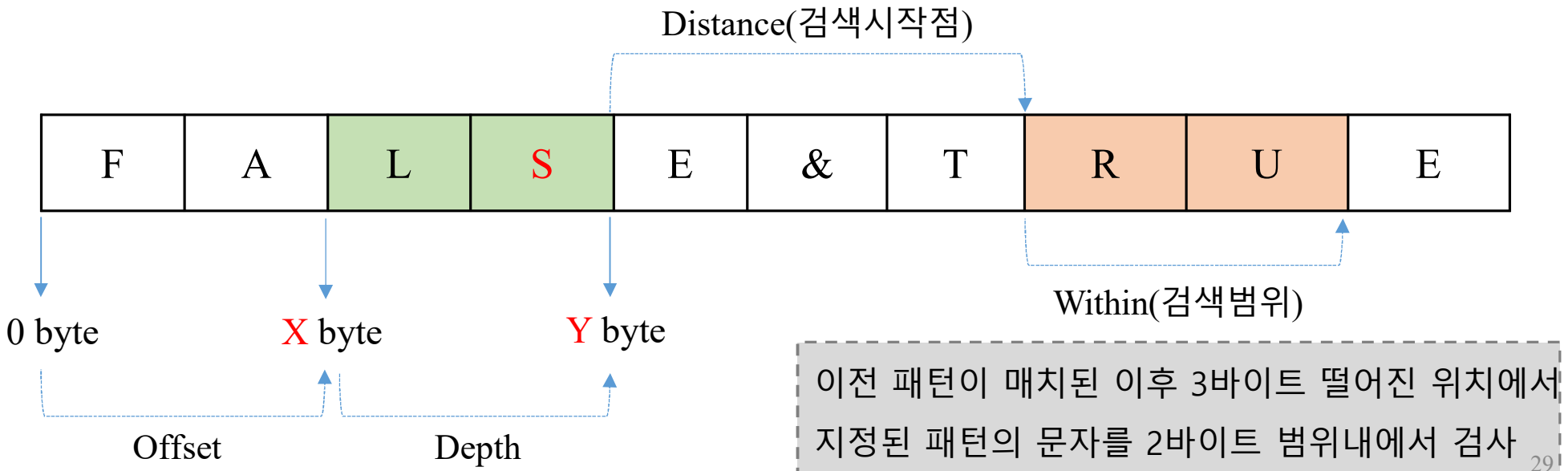
# 2) Detection Rule Example

## ❶ Detection Rule

alert tcp $EXTERNAL_NET any → $HOME_NET any

(msg: " TEST"; content: "S"; offset:2; depth:2; content: "R"; distance:3; within:2;
sid:1000001;)

- 3번째 byte 부터 2byte 범위에 S 패턴이 있는 지 검사, 패턴이 검사된 이 후 3byte 떨어진 위치에서 2byte 범위 내에서 R이라는 문자가 있는지 검색



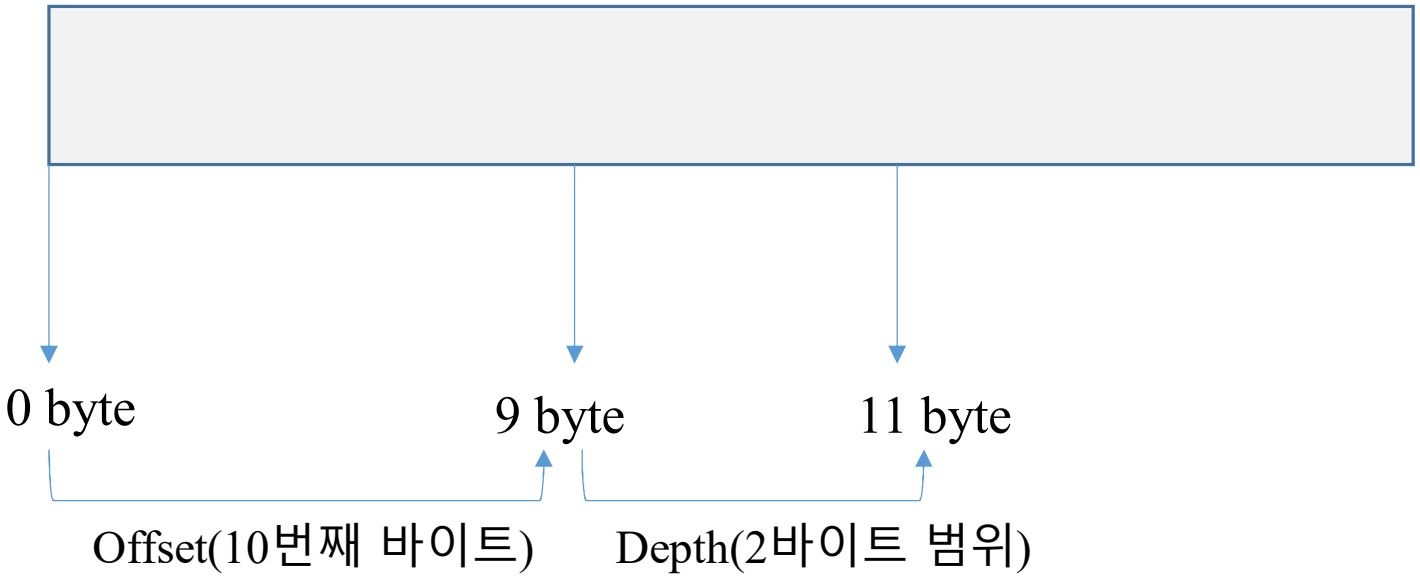

이전 패턴이 매치된 이후 3바이트 떨어진 위치에서 지정된 패턴의 문자를 2바이트 범위내에서 검사

## ❷ Detection Rule

alert tcp $EXTERNAL_NET any ⟶ $HOME_NET any

(msg: “ TEST”; content:|FFFF|; offset:9; depth:2; sid:1000001;)

- 10번째 byte 부터 2byte 범위에 FFFF 바이너리 패턴이 있는지 검사



## ❸ Detection Rule

```
alert tcp  any any ⟶ any 80
        (msg: “ Web Scan Detected”; content: “/administrator”;)
```

- 전송되는 패킷의 내용을 검사하여 “/administrator”란 문자열이 포함 된 경우 “Web Scan Detected”란 메시지로 로깅

## ❹ Detection Rule

```
alert tcp any any ⟶ any 80 (content: “root”; nocase;)
```

- 목적지 포트가 80인 모든 TCP 패킷에 대하여 대/소문자 구분없이 페이로드에 root문자열이 포함한 경우 alert 발생

## ❺ Detection Rule

```
alert tcp any any ⟶ any 22 (content: “login”; depth:10;)
```

- 목적지 포트가 22인 모두 TCP 패킷에 대하여 페이로드의 첫번째 byte부터 10byte 범위 내에 소문자 login 문자열이 포함한 경우 alert 발생

## ❻ Detection Rule

HTTP Flooding 공격 특징은 대부분 웹서버 공격 트래픽에서 최초 웹페이지에 대해 웹 접속 요청을 폭주시켜 세션자원을 소진검색 탐지 룰

```
alert tcp any any ⟶ any any (msg: “Get Flooding”; content:“Get / HTTP1.”; nocase;
depth:13; threshold:type threshold, track by_dst, count 10, seconds 1; sid:1000999)
```

- 첫 번째 바이트부터 13번째 바이트 범위 내에서 검색
  (offset을 명시하지 않으면 첫번째 byte부터 검색)
- Get : http request line, / : 호스트의 default page , HTTP1. : HTTP 버전
- 목적지 IP주소를 기준으로 1초마다 10번째 이벤트마다 alert action을 수행시켜 과도하게 많은 alert event가 발생하는 것을 방지

## ❼ Detection Rule

```
drop tcp any any ⟶ any any (msg: “ SYN/FIN Drop”; flags:SF;)
```

- 제어 플래그 중 SYN와 FIN이 동시에 설정되어 있는 TCP 패킷 차단한다.
- SYN은 연결 요청, FIN는 연결 종료를 위한 플래그이므로 동시에 설정될 수 없는 비정상 패킷이다. 비정상 패킷은 IDS/IPS의 탐지를 우회하여 공격 또는 스캐닝 을 위한 목적으로 사용되므로 이를 탐지 및 차단해야 한다.

## ❽ Detection Rule

```
alert tcp any any ⟶ any any (pcre: “ /POST.*Content\x2dLength\x3a\x20evilstring/”;)
```

- 목적지 주소 및 포트가 모두 any로 설정으로 모든 패킷을 검사한다. 이것은 장비에 많은 부하를 발생시킨다.
- HTTP 서비스를 제공하는 IP주소 및 PORT 정보를 파악해서 목적지 IP주소와 Port에 대한 검사만 검사룰을 적용시켜 장비의 부하를 줄일 수 있다.

## ❾ Detection Rule

```
alert tcp  any any   ⟶ any 80
( msg:“XSS Detect”;
 content:“GET”; offset:0; depth:3;
 content:“/login.php?id=%3Cscript%3E”; distance:1; sid:1000500 );
```

- 첫바이트~ 세번째 바이트 내에서 GET이라는 문자 검색
- 첫번째 매치된 문자열에서 1바이트 떨어진 곳에서부터 해당 문자 검색

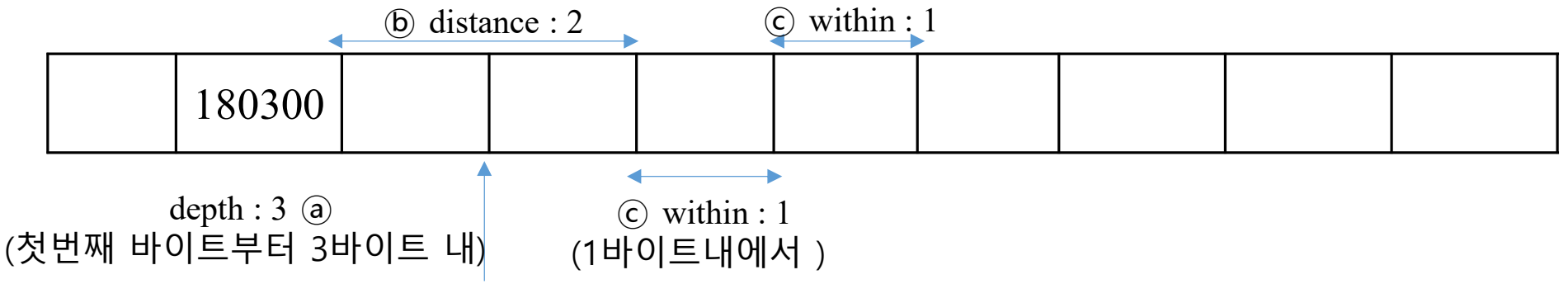
➔ Detection 이 안 될 경우 '대소문자' 옵션 첨부

# ❿ Detection Rule

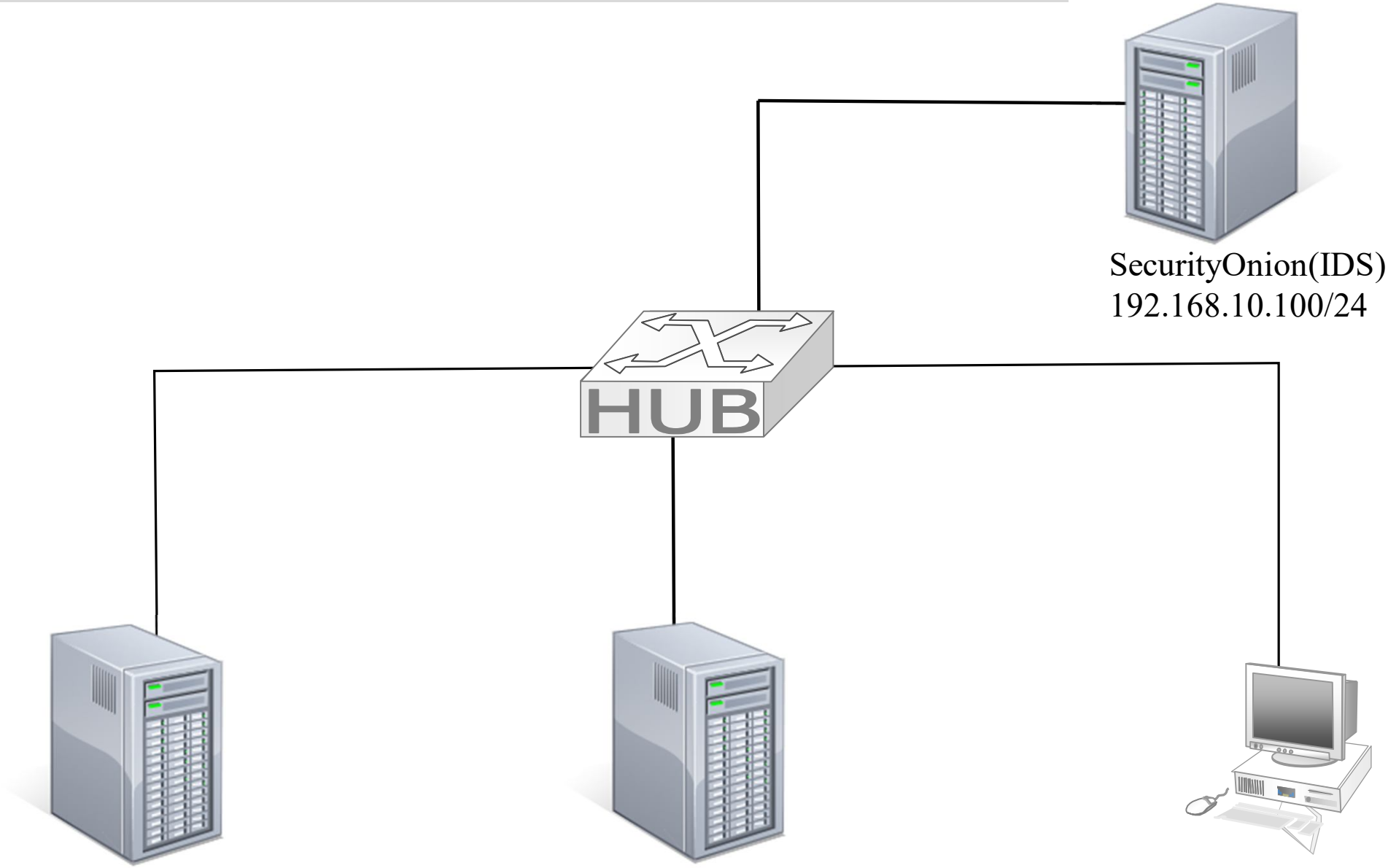
- OpenSSL 라이브러리의 하드비트(HeartBeat) 확장 모듈의 버그로 인해 발생하는 하트블리드(heartbleed) 취약점을 이용한 공격 탐지

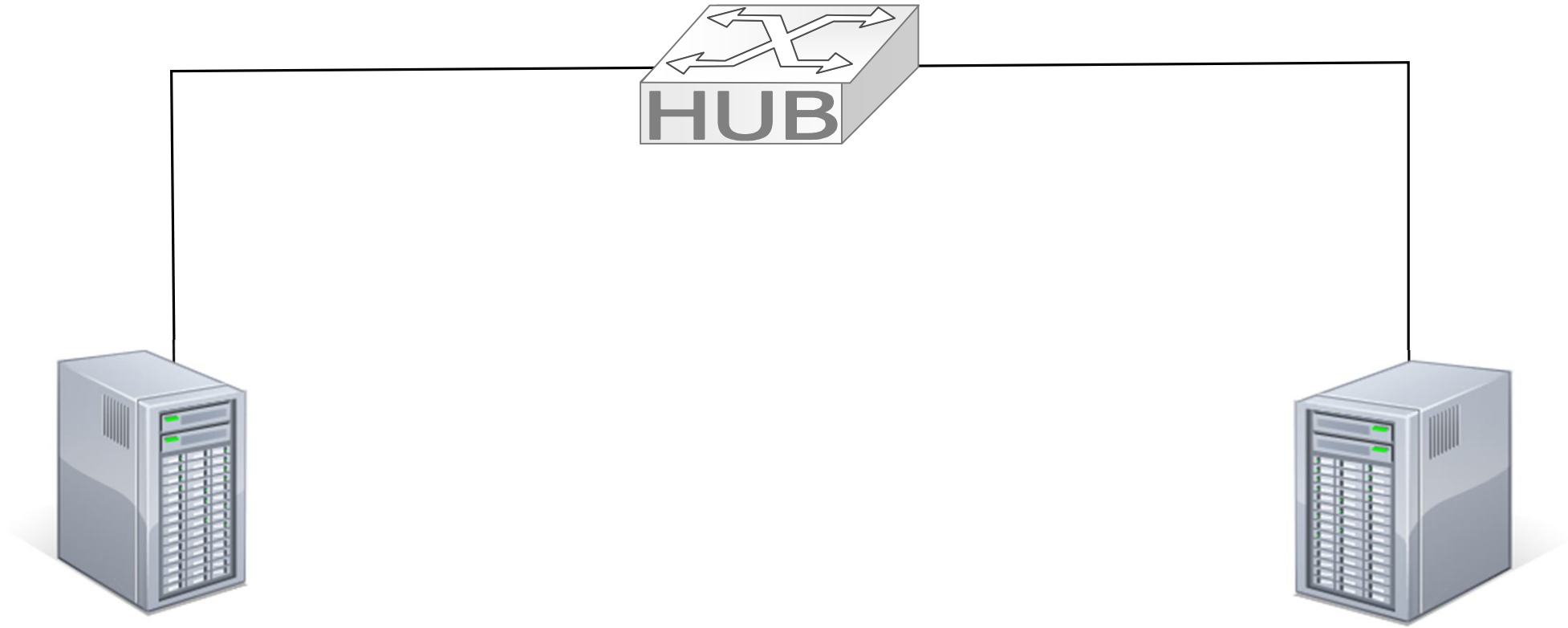
```
alert tcp any any <> any [443,465,563]
 (msg:"SSLv3 Malicious Heartbleed Request V2";
   content: "|18 03 00|"; depth:3;
   content: "|01|"; distance:2, within:1;
   content:! "|00|"; within:1; sid:100300;)
```

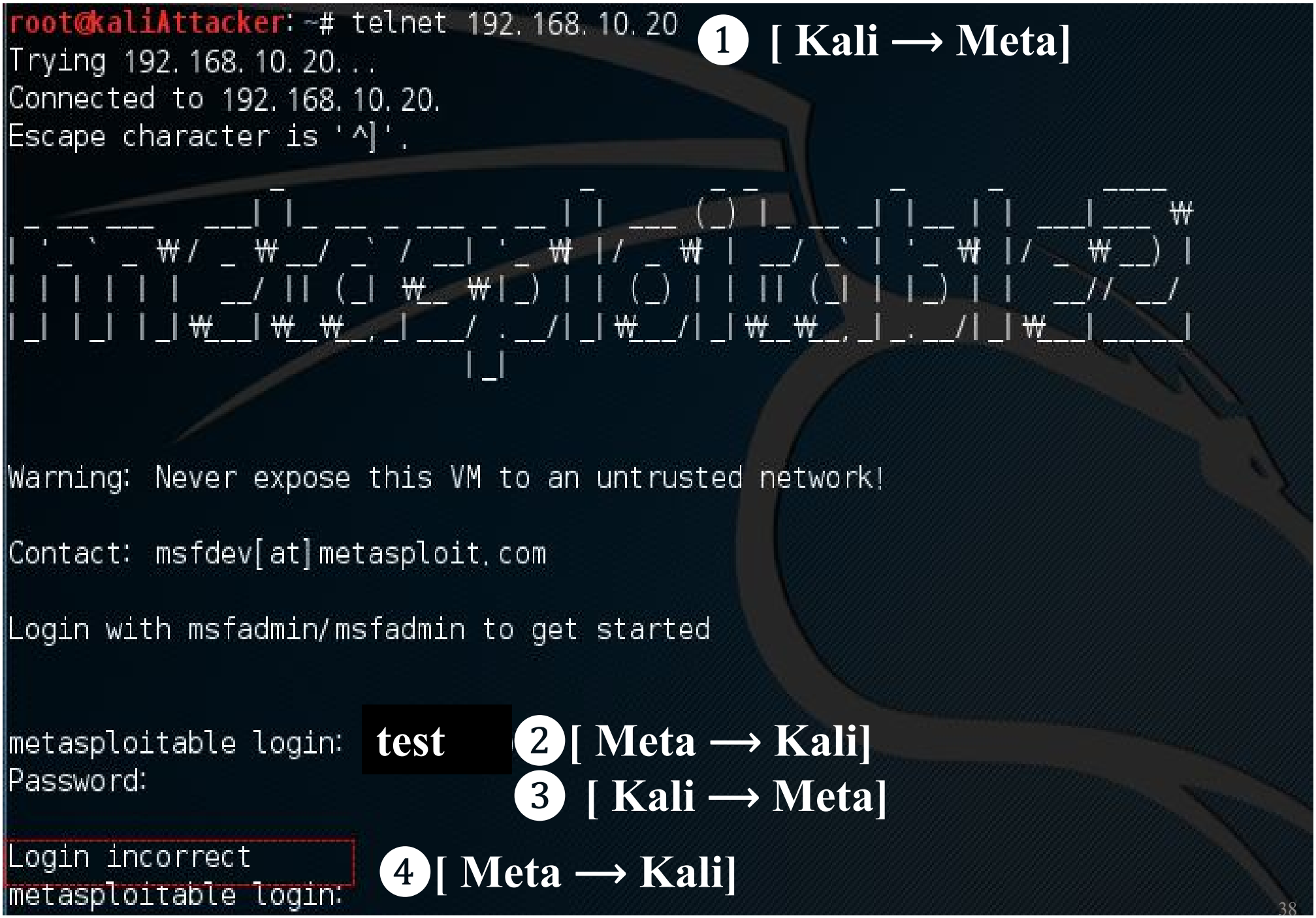
ⓐ 첫 바이트부터 3바이트 범위 내에서 패턴 감사
ⓑ 첫 번째 content가 매치 된 이후 2바이트 떨어진 위치에서 1바이트 내에서 지정된 패턴 검사
ⓒ 두 번째 content가 매치된 이후 1바이트 떨어진 위치에서 지정된 패턴 검사



# [실습] WebHacking Detect Rule 생성과 탐지

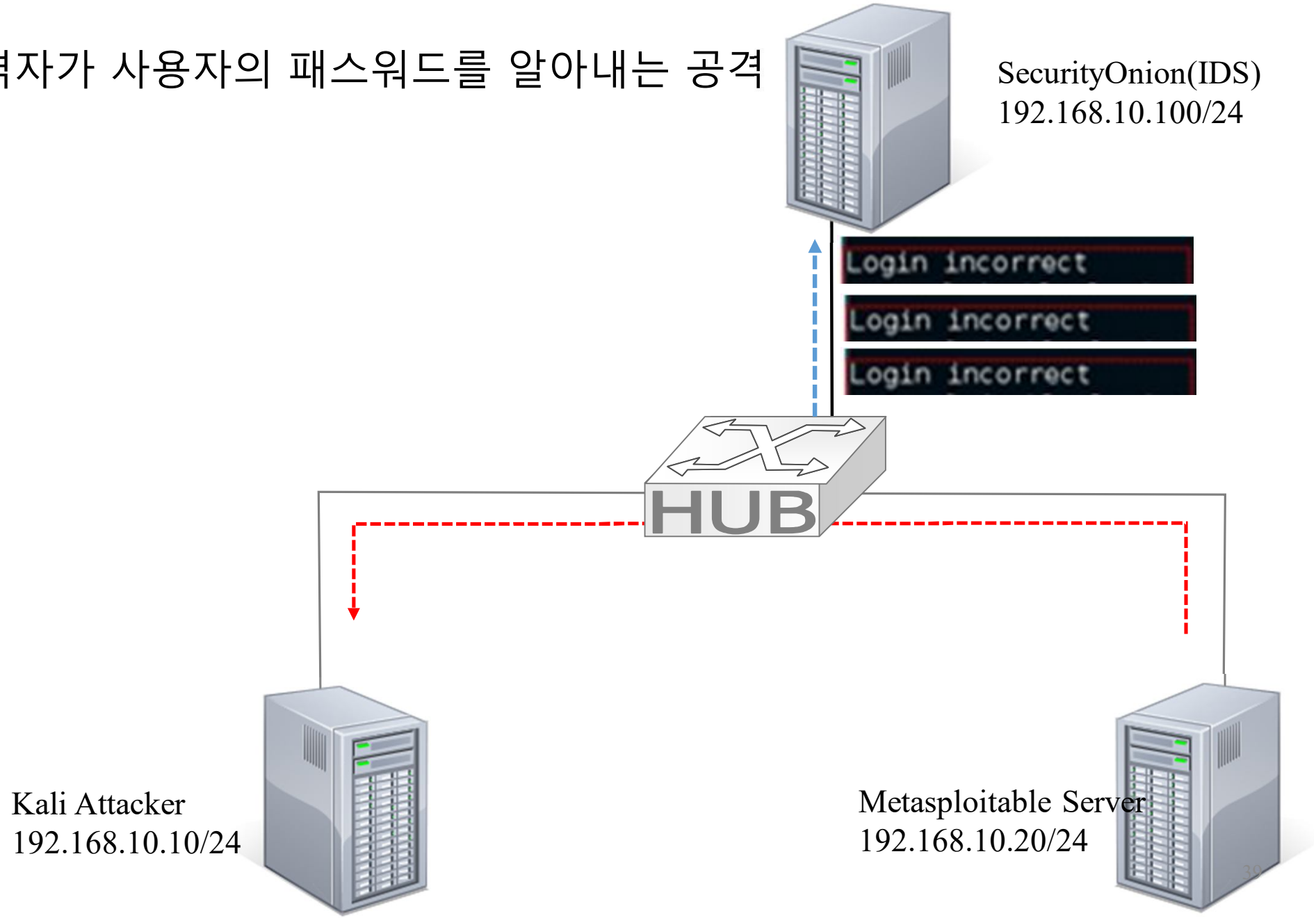



HUB
Kali Attacker
192.168.10.10/24
Metaspolitabe Server
192.168.10.20/24


root@kaliAttacker:~# telnet 192.168.10.20
Trying 192.168.10.20...
Connected to 192.168.10.20.
Escape character is '^]'.
Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started
metasploitable login:
test
Password:
Login incorrect
metasploitable login:
1 [ Kali → Meta]
2 [ Meta → Kali]
3 [ Kali → Meta]
4 [ Meta → Kali]

# 패스워드 크래킹 ( Password cracking )

- 공격자가 사용자의 패스워드를 알아내는 공격



# 공격 시그니처 찾기

(wireshark 패킷 수집기를 이용)

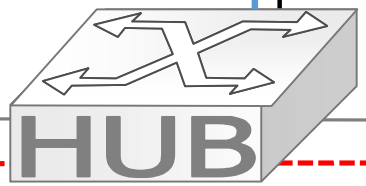
SecurityOnion(IDS)
192.168.10.200/24

Filter: telnet && ip.src == 192.168.10.20 Expression... Clear Apply 저장

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 69 | 41.10283300 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 72 | 41.27116100 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 75 | 41.43159200 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 78 | 41.54224900 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 81 | 41.78936700 | 192.168.10.20 | 192.168.10.10 | TELNET | 68 | Telnet Data ... |
| 83 | 41.78974600 | 192.168.10.20 | 192.168.10.10 | TELNET | 76 | Telnet Data ... |
| 99 | 43.12356500 | 192.168.10.20 | 192.168.10.10 | TELNET | 68 | Telnet Data ... |
| 103 | 45.50181000 | 192.168.10.20 | 192.168.10.10 | TELNET | 85 | Telnet Data ... |
| 105 | 45.51516200 | 192.168.10.20 | 192.168.10.10 | TELNET | 88 | Telnet Data ... |
| 111 | 48.79905100 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 114 | 48.91509700 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 118 | 49.06396700 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 121 | 49.11146100 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |
| 124 | 49.28038500 | 192.168.10.20 | 192.168.10.10 | TELNET | 67 | Telnet Data ... |

▸ Frame 103: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0
▸ Ethernet II, Src: Vmware_0b:16:25 (00:0c:29:0b:16:25), Dst: Vmware_5c:c3:e0 (00:0c:29:5c:c3:e0)
▸ Internet Protocol Version 4, Src: 192.168.10.20 (192.168.10.20), Dst: 192.168.10.10 (192.168.10.10)
▸ Transmission Control Protocol, Src Port: 23 (23), Dst Port: 36529 (36529), Seq: 696, Ack: 133, Len: 19
▾ Telnet
  Data: \r\n
  Data: Login incorrect\r\n

**패스워크 크래킹의 시그니처 : login incorrect**




Kali Attacker
192.168.10.10/24

Metasploitable
Server
192.168.10.20/24

# [SecurityOnion-snort] 탐지 정책 생성

❶ alert tcp 192.168.10.20/32 23 -> 192.168.10.10/32 any

(msg: "Telnet Fail"; content:"login incorrect"; nocase;

sid:3000001;)

❷ alert tcp 192.168.10.20/32 23 -> any  any

(msg: "Telnet Fail"; content:"login incorrect"; nocase;

id:3000001;)

# [SecurityOnion-snort] 탐지 정책 생성

❸ alert tcp 192.168.10.20/32 23 -> 192.168.10.10/32 any

(msg: "Telnet Attack";

**threshold:type both, track by_src, count 3, seconds 20**;

content:"login incorrect"; nocase; sid:3000004;)

- Threshold : 동일한 특정 패킷이 관리자 설정한 시간안 에 일정 수가 발견이 되면 경고 알림을 출력해주는 것.
- **threshold:type [limit,threshold,both], track [by_src, by_dst], count [몇초], seconds [횟수]**

limit : count 동안 횟 수번째 트래픽까지 탐지
threshold : 횟수 마다 계속 탐지
both : count 동안 횟수 만큼 트래픽이 탐지 될 시1번 만 탐지
by_src : 출발지 패킷만 해당
by_dst : 도착지 패킷만 해당

# Threshold

| 로그 발생 타입 | 로그 발생 기준 | 로그 발생 예시 |
|---|---|---|
| threshold:type **threshold**,<br>count 100, seconds 2; | 패킷양 | 2초내에 패킷 100개 : 로그 1개<br>2초내에 패킷 200개 : 로그 2개<br>4초내에 패킷 400개 : 로그 4개 |
| threshold:type **limit**,<br>count 100, seconds 2; | 임계시간 | 2초내에 패킷 100개 : 로그 1개<br>2초내에 패킷 200개 : 로그 1개<br>4초내에 패킷 400개 : 로그 2개 |
| threshold:type **both**,<br>count 100, seconds 2; | IP | 2초내에 패킷 100개 : 로그 1개<br>2초내에 패킷 200개 : 로그 1개<br>4초내에 패킷 400개 : 로그 1개 |