Json 파일에 수집되는 모든 로그는 통상적으로 다음과 같은 구조로 이루어진다.


```
{
  "timestamp": "2017-04-07T22:24:37.251547+0100",
  "flow_id": 586497171462735,
  "pcap_cnt": 53381,
  "event_type": "alert",
  "src_ip": "192.168.2.14",
  "src_port": 50096,
  "dest_ip": "209.53.113.5",
  "dest_port": 80,
  "proto": "TCP",
  "metadata": {
    "flowbits": [
      "http.dottedquadhost"
    ]
  },
  "tx_id": 4,
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2018358,
    "rev": 10,
    "signature": "ET HUNTING GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser
1",
    "category": "Potentially Bad Traffic",
    "severity": 2
  },
  "app_proto": "http"
}
```


**timestamp**:

- `"timestamp": "2017-04-07T22:24:37.251547+0100"`
- 이벤트가 발생한 시간과 날짜를 나타냅니다. 여기서는 2017년 4월 7일 22시 24분 37초에 발생했음을 알 수 있습니다. "+0100"은 시간대(UTC+1)를 나타냅니다.

**flow_id**:

- `"flow_id": 586497171462735`
- 특정 네트워크 흐름(Flow)을 식별하는 고유 ID입니다. 이 ID는 관련 패킷들이 동일한 세션 또는 연결에 속함을 나타냅니다.

**pcap_cnt**:

- `"pcap_cnt": 53381`
- pcap 파일에서 이 패킷이 몇 번째 패킷인지를 나타냅니다. 예를 들어, Wireshark와 같은 도구에서 이 번호를 사용해 해당 패킷을 쉽게 찾을 수 있습니다.

**event_type**:

- `"event_type": "alert"`
- 이 필드는 이벤트의 종류를 나타냅니다. 여기서는 "alert"(경고) 이벤트입니다.

**src_ip**:

- `"src_ip": "192.168.2.14"`
- 소스 IP 주소를 나타냅니다. 이 주소에서 트래픽이 시작되었습니다.

**src_port**:

- `"src_port": 50096`
- 소스 포트 번호를 나타냅니다. 이 포트는 소스 IP에서 사용된 포트 번호입니다.

**dest_ip**:

- `"dest_ip": "209.53.113.5"`
- 목적지 IP 주소를 나타냅니다. 이 주소로 트래픽이 전송되었습니다.

**dest_port**:

- `"dest_port": 80`
- 목적지 포트 번호를 나타냅니다. 이 경우, HTTP 트래픽이 주로 사용되는 80번 포트입니다.

**proto**:

- `"proto": "TCP"`
- 사용된 전송 프로토콜을 나타냅니다. 여기서는 TCP 프로토콜이 사용되었습니다.

**metadata**:

- `"metadata": { "flowbits": ["http.dottedquadhost"] }`
- 이벤트와 관련된 추가 정보를 포함합니다. 이 경우, `flowbits`는 특정 흐름이나 패턴을 나타내며, `"http.dottedquadhost"`라는 플로우 비트가 설정되어 있음을 나타냅니다.

**tx_id**:

- `"tx_id": 4`
- 이 트랜잭션의 고유 ID입니다. 같은 흐름 내에서의 특정 트랜잭션을 식별하는 데 사용됩니다.

**alert**:

- `"alert": { ... }`
- 경고에 대한 세부 정보를 포함하는 객체입니다.
  - **action**: `"allowed"` - 패킷이 허용되었음을 나타냅니다.
  - **gid**: `"gid": 1` - 규칙 그룹 ID를 나타냅니다.
  - **signature_id**: `"si gnature_id": 2018358` - 이벤트를 트리거한 규칙의 ID입니다.
  - **rev**: `"rev": 10` - 규칙의 개정 번호를 나타냅니다.
  - **signature**: `"signature": "ET HUNTING GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1"` - 규칙에 정의된 서명을 설명하는 문자열입니다.
  - **category**: `"category": "Potentially Bad Traffic"` - 이벤트의 카테고리를 나타냅니다.
  - **severity**: `"severity": 2` - 이벤트의 심각도 수준을 나타냅니다.

**app_proto**:

- `"app_proto": "http"`
- 감지된 애플리케이션 계층 프로토콜을 나타냅니다. 여기서는 HTTP 프로토콜이 사용되었습니다.

이 로그는 Suricata가 트래픽 중 특정 서명 규칙에 맞는 패킷을 감지했음을 나타내며, 관련된 세부 정보들을 포함하고 있습니다. 보안 분석 시 중요한 정보를 제공하여 어떤 트래픽이 경고를 발생시켰는지 파악하는 데 도움이 됩니다.

## Flow_id

```
$ jq 'select(.flow_id==1676750115612680)' eve.json
```

이 예시에서는 특정 `flow_id` 값을 가진 모든 로그 항목을 필터링하여 관련된 모든 이벤트를 쉽게 확인할 수 있습니다.

`flow_id`는 보안 관점에서 매우 중요한 필드입니다. 그 이유는 다음과 같습니다:

1. 이벤트 상관 관계: `flow_id`를 통해 다양한 이벤트(예: HTTP 트래픽, 경고, 파일 트랜잭션 등)를 동일한 네트워크 세션이나 플로우와 연결할 수 있습니다. 이를 통해 특정 세션에서 발생한 모든 활동을 추적하고 분석할 수 있습니다.
2. 침해 분석: 보안 침해가 의심되는 경우, `flow_id`를 사용하여 해당 세션에서 발생한 모든 관련 이벤트를 식별하고, 이를 기반으로 공격 경로를 재구성하거나 이상 활동을 확인할 수 있습니다.

3. 조사 효율성: 동일한 `flow_id`를 가진 이벤트를 한꺼번에 필터링할 수 있기 때문에, 복잡한 네트워크 환경에서 로그를 분석할 때 조사 효율성을 크게 향상시킬 수 있습니다.
4. 종합적인 분석: `flow_id`는 여러 가지 로그를 통합하여 전체 네트워크 흐름을 파악하는 데 도움을 줍니다. 이는 개별 이벤트를 단순히 보는 것보다 더 깊이 있는 분석을 가능하게 합니다.

따라서 `flow_id`는 네트워크 보안에서 이벤트를 상관시키고 종합적으로 분석하는 데 중요한 역할을 하므로, 보안 관점에서 매우 중요한 필드로 간주됩니다.

## PCAP fields

이 두 가지 필드는 네트워크 보안 분석에서 매우 중요한 역할을 합니다. 아래에 초등학생도 이해할 수 있는 수준으로 쉽게 설명해 볼게요.

**1. `pcap_cnt` 필드**

- 이게 뭐야**?**
  이 숫자는 PCAP 파일에서 몇 번째 패킷(데이터 조각)인지를 나타내요.
- 어떻게 사용해**?**
  예를 들어, 학교에서 큰 책을 읽고 있는데, 선생님이 "123페이지를 봐라"라고 하면, 바로 123페이지로 갈 수 있죠?.
  보안 전문가들은 이 숫자를 보고 특정 문제를 일으킨 네트워크 데이터를 빠르게 찾을 수 있어요.

**2. `pcap_filename` 필드**

- 이게 뭐야**?**
  이 필드는 어떤 파일에서 문제가 생겼는지 그 파일의 이름과 위치를 알려줘요.
- 어떻게 사용해**?**
  예를 들어, 너희 집에 있는 모든 책 중에서 선생님이 갑자기 "노란 책의 45페이지를 봐라"라고 하면, 너는 책장에 가서 노란 책을 꺼내 45페이지를 찾겠죠?
  이 `pcap_filename`은 그 노란 책이 어디에 있는지 알려주는 거예요. 그래서 전문가들이 파일을 쉽게 찾아 문제를 분석할 수 있게 해 줘요.

왜 이게 중요한가**?**

- 보안 전문가들은 때때로 네트워크에서 문제가 생긴 이유를 찾기 위해 데이터를 조사해야 해요. 이 두 필드를 사용하면 그 문제가 발생한 데이터를 정확히 찾아내서, 어떤 일이 벌어졌는지 알아낼 수 있어요.
- 예를 들어, 컴퓨터가 느려진 이유를 알고 싶을 때, `pcap_cnt`와 `pcap_filename`을 통해 "아, 이 데이터가 문제를 일으켰구나!"라고 쉽게 알 수 있죠.

이렇게 설명하면 `pcap_cnt`와 `pcap_filename` 필드가 얼마나 중요한지 이해하기 쉬울 거예요!

**1.** 악성 소프트웨어**(**멀웨어**)** 감염

- 문제 상황**:**
  어느 날 회사의 한 컴퓨터가 갑자기 이상하게 느려지거나, 알 수 없는 팝업창이 계속 뜬다면, 이는 악성 소프트웨어에 감염된 징후일 수 있어요.
- 이 필드들이 어떻게 도움을 줄 수 있을까**?**
  보안 전문가는 `pcap_cnt`와 `pcap_filename` 필드를 사용해서 악성 소프트웨어가 처음 네트워크에 들어온 순간의 데이터를 찾아낼 수 있어요. 이를 통해 어떤 파일이 네트워크를 통해 들어왔는지, 그리고 그 파일이 어떤 행동을 했는지 추적할 수 있어요.

**2.** 데이터 유출

- 문제 상황**:**
  회사에서 중요한 정보가 외부로 유출되는 사고가 발생했다고 가정해볼게요. 누군가가 중요한 문서를 인터넷을 통해 외부로 보낸 거예요.
- 이 필드들이 어떻게 도움을 줄 수 있을까**?**
  `pcap_cnt`와 `pcap_filename` 필드를 통해 보안 전문가는 해당 문서가 전송된 정확한 순간을 찾아낼 수 있어요. 이를 통해 누가, 언제, 어디로 데이터를 보냈는지 추적할 수 있어요. 이렇게 하면 데이터가 유출된 경로를 알아내고 추가적인 유출을 막을 수 있죠.

**3.** 네트워크 공격 **(**예**: DDoS** 공격**)**

- 문제 상황**:**
  어떤 웹사이트가 갑자기 엄청나게 느려지거나 아예 접속이 안 되는 경우가 있어요. 이는 DDoS(디도스) 공격이라고 불리는, 해커들이 여러 대의 컴퓨터를 이용해 한꺼번에 서버를 공격하는 경우에 발생할 수 있어요.
- 이 필드들이 어떻게 도움을 줄 수 있을까**?**
  보안 전문가는 `pcap_cnt`와 `pcap_filename` 필드를 사용해, 어떤 IP 주소에서 공격이 시작되었는지, 몇 개의 패킷이 공격에 사용되었는지 정확히 알아낼 수 있어요. 이를 통해 공격을 차단하고, 방어책을 세울 수 있죠.

**4.** 사기성 이메일**(**피싱**)**

- 문제 상황**:**
  직원이 이메일을 통해 사기성 링크를 클릭하게 되고, 그 링크가 악성 웹사이트로 연결되는 경우가 있어요. 이로 인해 네트워크에 악성 소프트웨어가 설치될 수 있어요.
- 이 필드들이 어떻게 도움을 줄 수 있을까**?**
  보안 전문가는 `pcap_cnt`와 `pcap_filename`을 사용해, 해당 이메일이 처음 도착한

순간과 클릭된 링크의 데이터를 분석할 수 있어요. 이를 통해 어떤 링크가 위험했는지, 그로 인해 어떤 데이터가 전송되었는지 알아낼 수 있어요.

이와 같은 문제들은 보안 전문가들이 해결해야 할 다양한 상황 중 일부예요. `pcap_cnt`와 `pcap_filename` 필드는 이러한 문제들을 정확히 분석하고 해결하는 데 매우 유용해요.

---

---

---

## Event Type: Alert


```
{
  "timestamp": "2023-09-18T06:13:41.532140+0000",
  "flow_id": 1676750115612680,
  "pcap_cnt": 130,
  "event_type": "alert",
  "src_ip": "142.11.240.191",
  "src_port": 35361,
  "dest_ip": "192.168.100.237",
  "dest_port": 49175,
  "proto": "TCP",
  "pkt_src": "wire/pcap",
  "ether": {
    "src_mac": "52:54:00:36:3e:ff",
    "dest_mac": "12:a9:86:6c:77:de"
  },
  "tx_id": 1,
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2045001,
    "rev": 1,
    "signature": "ET ATTACK_RESPONSE Win32/LeftHook Stealer Browser Extension Config
Inbound",
    "category": "A Network Trojan was detected",
    "severity": 1,
    "metadata": {
      "affected_product": [
        "Windows_XP_Vista_7_8_10_Server_32_64_Bit"
      ],
      "attack_target": [
        "Client_Endpoint"
      ],
      "created_at": [
        "2023_04_17"
      ],
      "deployment": [
```

```
        "Perimeter"
      ],
      "former_category": [
        "ATTACK_RESPONSE"
      ],
      "signature_severity": [
        "Major"
      ],
      "updated_at": [
        "2023_04_18"
      ]
    }
  },
  "http": {
    "hostname": "142.11.240.191",
    "http_port": 35361,
    "url": "/",
    "http_content_type": "text/xml",
    "http_method": "POST",
    "protocol": "HTTP/1.1",
    "status": 200,
    "length": 5362
  },
  "files": [
    {
      "filename": "/",
      "gaps": false,
      "state": "CLOSED",
      "stored": false,
      "size": 5362,
      "tx_id": 1
    }
  ],
  "app_proto": "http",
  "direction": "to_client",
  "flow": {
    "pkts_toserver": 13,
    "pkts_toclient": 12,
    "bytes_toserver": 1616,
    "bytes_toclient": 8044,
    "start": "2023-09-18T06:13:33.324862+0000",
    "src_ip": "192.168.100.237",
    "dest_ip": "142.11.240.191",
    "src_port": 49175,
    "dest_port": 35361
  }
}
```


이 로그는 Suricata에서 감지된 보안 경고(`alert`)를 나타냅니다. 주요 보안 포인트를 간략히 설명하겠습니다:

## 주요 내용:

- **flow_id**: 1676750115612680로, 이 세션의 모든 이벤트를 추적할 수 있습니다.
- 경고 내용 **(alert)**:
    - 서명: "Win32/LeftHook Stealer"라는 트로이목마가 탐지되었습니다.
    - 카테고리: 네트워크 트로이 목마가 감지되었음을 나타냅니다.
    - 심각도: 1로, 매우 심각한 보안 위협을 의미합니다.
    - 목표: 클라이언트 엔드포인트(사용자 PC)가 공격 대상입니다.
- 네트워크 정보:
    - 출발지 **IP**: 142.11.240.191 (외부 공격자)
    - 목적지 **IP**: 192.168.100.237 (내부 네트워크)
    - 프로토콜: TCP, HTTP 통신을 사용.
- **HTTP** 정보:
    - 요청 **URL**: /로, 공격자가 POST 요청을 통해 악성 데이터를 전송한 것으로 보입니다.
    - 응답 상태: 200, 정상적인 HTTP 응답이 반환되었습니다.
    - 파일 정보: 5362 바이트 크기의 파일이 전송되었습니다.

### 보안 의미**:**

이 로그는 외부 공격자가 내부 네트워크의 클라이언트를 목표로 트로이목마를 전송한 흔적을 나타냅니다. 이 경고는 즉각적인 주의와 대응이 필요함을 시사하며, 해당 세션을 기반으로 추가적인 조사와 방어 조치가 필요합니다.

## Action Field

```
"action":"allowed"
```

**"allowed"**:
이 값은 패킷이나 흐름이 허용되었음을 의미해요. 즉, 보안 규칙에 의해 차단되지 않고 네트워크를 통해 전송될 수 있었다는 뜻이에요. 예를 들어, 정상적인 웹 트래픽이나 허용된 네트워크 활동이 이 상태에 속해요.
**"blocked"**:
이 값은 패킷이나 흐름이 차단되었음을 의미해요. 즉, 보안 규칙에 의해 중지되거나 거부되었고, 네트워크를 통해 더 이상 전송되지 않았다는 뜻이에요. 이는 해킹 시도나 악의적인 트래픽일 가능성이 있어요.

## Verdict

```
"verdict": {
   "action": "drop",
   "reject-target": "to_client",
   "reject": "[icmp-prohib]"
 }
```

**Verdict** 필드는 특정 패킷에 대해 최종적으로 적용될 조치를 나타내는 정보를 담고 있고, 패킷에 대해 발생한 모든 보안 규칙(서명)과 기타 이벤트(예: 흐름 차단)에 기반하여 결정돼요.

**Verdict**는 최종적으로 해당 패킷에 어떤 조치가 취해졌는지를 명확히 보여줘요.

구성 요소**:**

1. **action**:
    - `alert`, `pass`, `drop`
    - `drop`은 특히 IPS(침입 방지 시스템) 모드에서만 발생하며, 패킷이 차단되었음을 의미해요.
    - 예를 들어, 패킷이 어떤 규칙에 의해 허용(`allowed`)되었지만, 다른 규칙에 의해 차단(`drop`)될 수 있어요. 이 경우, **Verdict**는 `drop`으로 나타날 수 있어요.
2. **reject-target**:
    - 이 필드는 "reject" 규칙이 적용될 때 나타나며, 어떤 방향으로 패킷이 거부되었는지를 나타내요.
    - 가능한 값으로는 `to_server`, `to_client`, `both`가 있어요.
    - 예를 들어, `to_client`는 클라이언트로 향하는 트래픽이 거부되었음을 의미해요.
3. **reject**:
    - 이 필드는 "reject" 규칙이 적용된 경우에만 나타나며, 거부 타입을 배열 형태로 표시해요.
    - 가능한 값으로는 `tcp-reset`(TCP 연결 재설정)과 `icmp-prohib`(ICMP로 금지됨) 등이 있어요.

## Event Type: Anomaly

Anomaly 이벤트는 비정상적인 상황을 보고하는 이벤트로, 예를 들어 패킷이 잘리거나, 잘못된 값을 포함하거나, 추가 처리가 불가능한 패킷, 또는 예상치 못한 동작을 하는 경우가 해당됩니다.

네트워크에서 이러한 이상 현상이 자주 발생하면, anomaly 로깅이 활성화된 경우 패킷 처리 성능이 저하될 수 있습니다.

```
"anomaly": {
  "type": "decode",
  "event": "decoder.icmpv4.unknown_type"
}

"anomaly": {
  "type": "decode",
  "event": "decoder.udp.pkt_too_small"
}

"anomaly": {
  "type": "decode",
```

```
  "event": "decoder.ipv4.wrong_ip_version"
}

"anomaly": {
  "type": "stream",
  "event": "stream.pkt_invalid_timestamp"
}

{
  "timestamp": "1969-12-31T16:04:21.000000-0800",
  "pcap_cnt": 9262,
  "event_type": "anomaly",
  "src_ip": "208.21.2.184",
  "src_port": 0,
  "dest_ip": "10.1.1.99",
  "dest_port": 0,
  "proto": "UDP",
  "packet": "////////AQEBAQEBCABFAAA8xZ5AAP8R1+DQFQK4CgE=",
  "packet_info": {
    "linktype": 1
  },
  "anomaly": {
    "type": "decode",
    "event": "decoder.udp.pkt_too_small"
  }
}

{
  "timestamp": "2016-01-11T05:10:54.612110-0800",
  "flow_id": 412547343494194,
  "pcap_cnt": 1391293,
  "event_type": "anomaly",
  "src_ip": "192.168.122.149",
  "src_port": 49324,
  "dest_ip": "69.195.71.174",
  "dest_port": 443,
  "proto": "TCP",
  "app_proto": "tls",
  "anomaly": {
    "type": "applayer",
    "event": "APPLAYER_DETECT_PROTOCOL_ONLY_ONE_DIRECTION",
    "layer": "proto_detect"
  }
}

{
  "timestamp": "2016-01-11T05:10:52.828802-0800",
  "flow_id": 201217772575257,
  "pcap_cnt": 1391281,
  "event_type": "anomaly",
  "src_ip": "192.168.122.149",
  "src_port": 49323,
  "dest_ip": "69.195.71.174",
  "dest_port": 443,
  "proto": "TCP",
```

```
  "tx_id": 0,
  "app_proto": "tls",
  "anomaly": {
    "type": "applayer",
    "event": "INVALID_RECORD_TYPE",
    "layer": "proto_parser"
  }
}
```

1. **"type"** 필드:
    - 설명: 이벤트의 유형을 나타냅니다. 값은 "decode", "stream", "applayer" 중 하나일 수 있습니다. 드물게 "unknown"이 될 수도 있습니다. "unknown"인 경우 "code" 필드가 추가로 포함됩니다.
    - 사용 사례: "applayer" 타입의 이벤트는 애플리케이션 계층 파서가 감지한 것입니다.
2. **"event"** 필드:
    - 설명: 이상 이벤트의 이름을 나타냅니다. "decode" 유형의 이벤트는 "decoder"로 시작하며, "stream" 유형의 이벤트는 "stream"으로 시작합니다.
    - 사용 사례: 어떤 종류의 이상 현상이 발생했는지 명확히 파악할 수 있습니다.
3. **"code"** 필드:
    - 설명: "type"이 "unknown"일 때만 포함되며, 인식되지 않은 이벤트 코드가 들어 있습니다. 그렇지 않으면 이 필드는 존재하지 않습니다.
    - 사용 사례: 비정상적인 이벤트에 대해 추가적인 디버깅 정보를 제공하는 데 유용합니다.
4. **"layer"** 필드:
    - 설명: "type"이 "applayer"일 때 포함되며, 이벤트를 감지한 계층을 나타냅니다. 값은 "proto_parser" (프로토콜 파서), "proto_detect" (프로토콜 감지), 또는 "parser"일 수 있습니다.
    - 사용 사례: 어떤 레이어에서 문제가 발생했는지 추적하는 데 유용합니다.
5. **"packethdr"** 필드:
    - 설명: 이 옵션이 활성화되면, 패킷의 첫 32바이트가 base64로 인코딩되어 기록됩니다. 이는 "type"이 "packet" 또는 "stream"일 때만 해당됩니다.
    - 사용 사례: 특정 패킷의 헤더를 분석하여 무엇이 잘못되었는지 더 깊이 이해하는 데 유용합니다.

예시 사용 사례: 만약 네트워크에서 계속해서 이상한 패킷들이 감지된다면, 이런 anomaly 이벤트 로그를 통해 어떤 종류의 패킷이 문제를 일으키고 있는지 파악하고, 이를 해결하기 위한 조치를 취할 수 있습니다.

## Event type: HTTP

기본 필드:

1. **hostname**: 이 HTTP 이벤트와 관련된 호스트 이름.
2. **url**: 해당 호스트에서 접근된 URL.
3. **http_user_agent**: 요청을 보낸 소프트웨어(예: 웹 브라우저)를 나타내는 사용자 에이전트 문자열.
4. **http_content_type**: 반환된 데이터의 종류 (예: `application/x-gzip`).
5. **cookie**: HTTP 요청과 관련된 쿠키.

확장 로그 필드**:**

`suricata.yaml` 설정 파일에서 확장 로깅이 활성화된 경우 추가로 기록되는 필드:

1. **length**: HTTP 본문의 내용 크기.
2. **status**: HTTP 상태 코드 (예: 200, 404).
3. **protocol**: 사용된 HTTP 프로토콜 버전 (예: HTTP/1.1).
4. **http_method**: 사용된 HTTP 메서드 (예: GET, POST).
5. **http_refer**: 요청이 시작된 참조 URL.

# Event type: DNS

Suricata의 DNS 로깅은 네트워크에서 오가는 DNS 요청과 응답을 기록하여 보안을 강화하는 데 중요한 역할을 합니다. 쉽게 말해, DNS는 인터넷 주소를 찾는 전화번호부 같은 역할을 하는데, Suricata는 이 과정에서 일어나는 모든 일을 꼼꼼히 기록해서 이상한 점이 없는지 확인합니다.

주요 포인트는 다음과 같습니다:

- **DNS** 메시지 종류: 이 기록을 통해 누가 어떤 주소를 찾았고, 그 결과가 무엇이었는지 알 수 있습니다.
- 플래그들: 예를 들어, `qr` 같은 플래그는 요청에 대한 응답이 제대로 왔는지, `aa`는 그 응답이 신뢰할 수 있는지 등을 보여줍니다.
- 응답 코드: `rcode`는 요청이 성공적으로 처리되었는지 아닌지를 기록합니다.

Suricata는 기록할 DNS 데이터의 종류를 설정할 수 있어, 보안 전문가들이 필요에 따라 특정 데이터만 집중해서 볼 수 있습니다. 이를 통해 네트워크에서 일어나는 이상한 행동이나 공격을 빨리 발견하고, 이를 막는 데 큰 도움이 됩니다.

```
- eve-log:
    enabled: yes
    type: file #file|syslog|unix_dgram|unix_stream
    filename: eve.json
    # the following are valid when type: syslog above
    #identity: "suricata"
    #facility: local5
    #level: Info ## possible levels: Emergency, Alert, Critical,
                 ## Error, Warning, Notice, Info, Debug
    types:
      - alert
      - dns:

        # Logging format. In 8.0 version 3 is the default. Can be
        # set to 2 to keep compatibility with Suricata 7.0.
        # version: 3

        # Control logging of requests and responses:
        # - requests: enable logging of DNS queries
        # - responses: enable logging of DNS answers
        # By default both requests and responses are logged.
        requests: yes
        responses: yes
        # DNS record types to log, based on the query type.
        # Default: all.
        #types: [a, aaaa, cname, mx, ns, ptr, txt]
        types: [a, ns, md, mf, cname, soa, mb, mg, mr, null,
        wks, ptr, hinfo, minfo, mx, txt, rp, afsdb, x25, isdn,
        rt, nsap, nsapptr, sig, key, px, gpos, aaaa, loc, nxt,
        srv, atma, naptr, kx, cert, a6, dname, opt, apl, ds,
        sshfp, ipseckey, rrsig, nsec, dnskey, dhcid, nsec3,
        nsec3param, tlsa, hip, cds, cdnskey, spf, tkey,
        tsig, maila, any, uri]
```

이 Suricata 설정 파일의 `eve-log` 섹션은 보안 관점에서 중요한 DNS 로깅 설정을 포함하고 있으며, 각 필드는 다음과 같은 역할을 합니다:

- **enabled**: `yes`로 설정되어 있어 로깅이 활성화됨을 의미합니다. 이를 통해 Suricata가 이벤트 로그를 기록하게 됩니다.
- **type**: 로깅의 출력 형식을 정의합니다. `file`로 설정되어 있어 로그가 파일(`eve.json`)로 저장됩니다. 이 파일은 보안 분석 시 중요한 데이터를 제공하게 됩니다.
- **filename**: 로그가 저장될 파일 이름을 지정합니다. `eve.json` 파일에 모든 로그가 저장되며, 이는 보안 이벤트 분석에 활용됩니다.

- **types**: Suricata가 로깅할 이벤트 유형을 지정합니다. 여기서는 `alert`와 `dns` 이벤트를 로깅하도록 설정되어 있습니다.
- **version**: DNS 로깅 형식의 버전을 지정합니다. 버전 3이 기본이며, Suricata 8.0의 최신 형식을 사용합니다. 이전 버전과의 호환성을 유지하려면 2로 설정할 수 있습니다.
- **requests**: `yes`로 설정되어 있으며, 이는 DNS 쿼리(요청)를 로깅하겠다는 의미입니다. 이를 통해 어떤 도메인이 쿼리되었는지를 추적할 수 있으며, 잠재적인 보안 위협을 식별할 수 있습니다.
- **responses**: `yes`로 설정되어 있으며, DNS 응답을 로깅합니다. 이를 통해 쿼리에 대한 응답을 분석하여 위협을 탐지할 수 있습니다.
- **types**: 로깅할 DNS 레코드 유형을 지정합니다. 설정된 다양한 레코드 유형(A, NS, MX 등)은 각기 다른 DNS 쿼리 유형을 다룹니다. 특정 레코드 유형을 모니터링하여 악성 DNS 활동이나 의심스러운 트래픽을 식별할 수 있습니다.

이 설정은 DNS 기반 공격이나 데이터 유출을 감지하는 데 중요한 역할을 합니다. 예를 들어, `requests`와 `responses`의 로깅을 통해, 의심스러운 도메인 쿼리와 응답을 추적하고 분석하여 공격자의 활동을 조기에 탐지할 수 있습니다. 또한, 다양한 레코드 유형을 로깅함으로써 DNS 인프라 전반에 대한 심층적인 가시성을 확보할 수 있습니다.

## Event type: FTP

Example of regular FTP logging:

```
"ftp": {
  "command": "RETR",
  "command_data": "100KB.zip",
  "reply": [
    "Opening BINARY mode data connection for 100KB.zip (102400 bytes).",
    "Transfer complete."
  ],
  "completion_code": [
    "150",
    "226"
  ],
```

Example showing all fields:

```
"ftp": {
  "command": "EPRT",
  "command_data": "|2|2a01:e34:ee97:b130:8c3e:45ea:5ac6:e301|41813|",
  "reply": [
    "EPRT command successful. Consider using EPSV."
  ],
  "completion_code": [
    "200"
  ],
  "dynamic_port": 41813,
  "mode": "active",
  "reply_received": "yes"
}
```

**"command"**: FTP 명령어.

**"command_data"**: 명령어에 포함된 데이터.

**"reply"**: 명령어에 대한 응답으로, 여러 줄일 수 있으며 배열 형식으로 제공됨.

**"completion_code"**: 3자리 완료 코드. 첫 번째 자리는 응답이 좋은지, 나쁜지, 또는 완료되지 않았는지를 나타냅니다. 이 필드도 배열 형식으로 제공되며, 여러 응답 줄에 맞는 여러 완료 코드가 포함될 수 있습니다.

**"dynamic_port"**: "PORT" 또는 "EPRT" 명령어가 사용된 경우, 이후 데이터 전송을 위해 설정된 동적 포트.

**"mode"**: FTP 연결 유형. 대부분의 연결은 "패시브"이지만, "액티브"일 수도 있음.

**"reply_received"**: 명령어에 대한 응답이 일치했는지를 나타냅니다. 일부 비정형적인 경우에는 명령어에 응답이 없을 수도 있음.

## Event type: FTP_DATA

```
"ftp_data": {
  "filename": "temp.txt",
  "command": "RETR"
}
```

- **"filename"**: `"temp.txt"` (관련된 파일의 이름)
- **"command"**: `"RETR"` (파일을 가져오는 명령어)

이 예시는 `RETR` 명령어가 `"temp.txt"` 파일과 연관되어 사용된 상황을 기록한 로그입니다.

## Event type: TLS

- **"subject"**: TLS 인증서의 주체 필드 (이 인증서가 누구를 위해 발행되었는지 나타냄).
- **"issuer"**: TLS 인증서의 발급자 필드 (이 인증서를 발급한 기관 또는 개인).
- **"session_resumed"**: 이 필드의 값이 `"true"`이면, TLS 세션이 세션 ID를 통해 재개된 것입니다. 이 필드가 나타나면 "subject"와 "issuer"는 나타나지 않습니다. 이는 TLS 인증서를 볼 수 없기 때문입니다.

확장 로깅이 활성화된 경우 추가되는 필드:

- **"serial"**: TLS 인증서의 일련번호.
- **"fingerprint"**: TLS 인증서의 (SHA1) 지문.
- **"sni"**: 클라이언트가 보낸 서버 이름 표시(SNI) 확장 정보.
- **"version"**: 사용된 SSL/TLS 버전.
- **"notbefore"**: TLS 인증서의 NotBefore 필드 (인증서가 유효하기 시작하는 날짜).
- **"notafter"**: TLS 인증서의 NotAfter 필드 (인증서가 만료되는 날짜).
- **"ja3"**: JA3 지문, JA3 해시와 JA3 문자열로 구성됨.
- **"ja3s"**: JA3S 지문, JA3S 해시와 JA3 문자열로 구성됨.
- **"ja4"**: TLS 클라이언트 지문에 대한 JA4 지문.
- **"client_alpns"**: ALPN(애플리케이션 계층 프로토콜 협상) 값의 문자열 배열 (클라이언트 측).
- **"server_alpns"**: ALPN 값의 문자열 배열 (서버 측).

**JA3**와 **JA4** 필드들은 **Suricata** 설정 파일에서 활성화해야 포함됩니다 **(**설정에서 **'app-layer.protocols.tls.ja3-fingerprints'**와 **'app-layer.protocols.tls.ja4-fingerprints'**를 **'yes'**로 설정**).**

사용자 정의 로깅이 활성화된 경우 추가되는 필드:

- **"certificate"**: TLS 인증서가 base64로 인코딩된 형태.
- **"chain"**: 전체 TLS 인증서 체인이 base64로 인코딩된 형태.

이 필드들은 TLS 세션과 관련된 중요한 정보를 기록하여, 보안 분석이나 문제 해결 시 유용한 데이터를 제공합니다.

## Event type: TFTP

- **"packet"**: 작업 코드를 나타냅니다. `"read"`(읽기), `"write"`(쓰기), 또는 `"error"`(오류) 중 하나일 수 있습니다.
- **"file"**: TFTP 프로토콜을 통해 전송되는 파일의 이름을 나타냅니다.
- **"mode"**: 전송 모드를 나타냅니다. `"octet"`, `"mail"`, 또는 `"netascii"` 중 하나일 수 있으며, 대소문자 조합이 가능합니다.

**TFTP** 로그 예시

```
"tftp": {
    "packet": "write",
    "file": "rfc1350.txt",
    "mode": "octet"
}
```

- **packet**: `"write"` (파일 쓰기 요청)
- **file**: `"rfc1350.txt"` (전송되는 파일 이름)
- **mode**: `"octet"` (파일을 바이트 단위로 전송)

이 예시는 TFTP 프로토콜을 통해 `"rfc1350.txt"` 파일을 `"octet"` 모드로 쓰기(`"write"`) 요청을 한 상황을 기록한 로그입니다.

# Event type: SMB


```
"smb": {
  "id": 1,
  "dialect": "unknown",
  "command": "SMB2_COMMAND_CREATE",
  "status": "STATUS_SUCCESS",
  "status_code": "0x0",
  "session_id": 4398046511201,
  "tree_id": 1,
  "filename": "atsvc",
  "disposition": "FILE_OPEN",
  "access": "normal",
  "created": 0,
  "accessed": 0,
  "modified": 0,
  "changed": 0,
  "size": 0,
  "fuid": "0000004d-0000-0000-0005-0000ffffffff"
}
```


SMB(서버 메시지 블록) 프로토콜은 네트워크 파일 및 프린터 공유를 위한 중요한 프로토콜입니다. SMB는 다양한 버전과 함께 발전해왔으며, 보안이 강화된 최신 버전들은 네트워크에서 안전한 자원 공유를 가능하게 합니다. NTLMSSP와 Kerberos 같은 인증 프로토콜과의 결합으로, SMB는 대규모 네트워크 환경에서도 안전하게 사용할 수 있는 효율적인 자원 관리 도구로 자리 잡고 있습니다. 보안이 중요한 현대의 네트워크 환경에서, SMB의 적절한 사용과 관리가 중요하며, 최신 버전을 사용하고 보안 설정을 강화하는 것이 필수적입니다

다음은 SMB(서버 메시지 블록) 이벤트 타입에 대한 필드 설명을 한글로 번역한 것입니다. SMB는 파일, 프린터, 포트 등의 네트워크 자원을 공유하기 위한 프로토콜입니다.

### 17.1.2.10.1. SMB 필드

- **"id" (**정수**)**: 내부 트랜잭션 ID입니다.
- **"dialect" (**문자열**)**: 협상된 프로토콜 방언을 나타냅니다. 만약 이 정보가 없으면 "unknown"으로 표시됩니다.
- **"command" (**문자열**)**: SMB 명령어 이름을 나타냅니다. 예를 들어, `SMB2_COMMAND_CREATE` 또는 `SMB1_COMMAND_WRITE_ANDX`와 같은 형식입니다.
- **"status" (**문자열**)**: 상태 문자열로, NT_STATUS나 DOS_ERR와 같은 다양한 형식일 수 있습니다.
- **"status_code" (**문자열**)**: 상태 코드를 16진수 문자열로 나타냅니다.
- **"session_id" (**정수**)**: SMB2 이상에서는 세션 ID, SMB1에서는 사용자 ID를 나타냅니다.

- **"tree_id" (**정수**)**: 트리 ID를 나타냅니다. 이 값은 특정 공유 자원(예: 디렉토리)과 관련이 있습니다.
- **"filename" (**문자열**)**: `CREATE`와 같은 명령어에 사용된 파일 이름입니다.
- **"disposition" (**문자열**)**: 요청된 파일 처리 방법을 나타냅니다. 예를 들어, `FILE_OPEN`, `FILE_CREATE`, `FILE_OVERWRITE` 등이 있습니다. 관련 문서를 참조하세요.
- **"access" (**문자열**)**: 파일이 어떻게 열렸는지를 나타냅니다. "normal" 또는 "delete on close"가 있으며, 이 필드는 변경될 수 있습니다.
- **"created"**, **"accessed"**, **"modified"**, **"changed" (**정수**)**: 파일의 생성, 접근, 수정, 변경된 시간 타임스탬프를 나타내며, Unix 에포크(1970년 1월 1일 00:00:00 UTC) 이후의 초 단위로 기록됩니다.
- **"size" (**정수**)**: 요청된 파일의 크기입니다.
- **"fuid" (**문자열**)**: SMB2 이상에서는 파일 GUID, SMB1에서는 FID를 16진수로 나타냅니다.
- **"share" (**문자열**)**: 공유 이름을 나타냅니다. 예를 들어, 공유된 폴더의 이름일 수 있습니다.
- **"share_type" (**문자열**)**: 공유의 유형을 나타냅니다. `FILE`, `PIPE`, `PRINT` 또는 "unknown"이 있을 수 있습니다.
- **"client_dialects" (**문자열 배열**)**: 클라이언트가 지원하는 SMB 방언(프로토콜 버전) 목록입니다.
- **"client_guid" (**문자열**)**: 클라이언트의 GUID(글로벌 고유 식별자)입니다.
- **"server_guid" (**문자열**)**: 서버의 GUID입니다.
- **"request.native_os" (**문자열**)**: SMB1에서 클라이언트의 네이티브 OS 문자열입니다.
- **"request.native_lm" (**문자열**)**: SMB1에서 클라이언트의 네이티브 Lan Manager 문자열입니다.
- **"response.native_os" (**문자열**)**: SMB1에서 서버의 네이티브 OS 문자열입니다.
- **"response.native_lm" (**문자열**)**: SMB1에서 서버의 네이티브 Lan Manager 문자열입니다.

이 필드들은 SMB 통신에서 발생하는 다양한 작업과 상태를 기록하여, 네트워크 파일 공유와 관련된 상세한 정보를 제공합니다. 보안 분석, 네트워크 모니터링, 문제 해결 시 유용하게 활용될 수 있습니다.

파이프 열기(`"SMB2_COMMAND_CREATE"`)는 클라이언트가 서버와의 특정 통신 채널(파이프)을 통해 데이터를 주고받기 위해 그 파이프를 열도록 요청하는 과정입니다. 성공적으로 열리면 클라이언트는 파이프를 통해 작업을 수행할 수 있습니다.

```
"smb": {
  "id": 15,
  "dialect": "2.10",
  "command": "SMB2_COMMAND_CLOSE",
  "status": "STATUS_SUCCESS",
  "status_code": "0x0",
  "session_id": 4398046511121,
  "tree_id": 1,
}
```


파이프 닫기("SMB2_COMMAND_CLOSE")는 이 파이프와의 통신이 완료된 후, 더 이상 필요하지 않게 된 파이프를 닫는 작업입니다. 이 작업이 성공적으로 완료되면, 해당 파이프에 대한 연결이 종료됩니다.

"Tree connect (share open)" 작업은 SMB 프로토콜에서 클라이언트가 서버의 특정 공유 자원(예: 공유 폴더)에 연결을 설정하는 과정을 의미합니다. 주어진 로그 예시를 바탕으로 이 작업의 의미를 상세히 설명하겠습니다.

### **Tree Connect (Share Open)**의 의미

- **Tree Connect**는 SMB에서 클라이언트가 서버의 특정 공유 자원(폴더, 파일, 프린터 등)에 접근하기 위해 연결을 설정하는 작업입니다. 이 과정은 클라이언트가 서버의 리소스에 접근하기 전에 필수적으로 수행됩니다.

```
"smb": {
  "id": 3,
  "dialect": "2.10",
  "command": "SMB2_COMMAND_TREE_CONNECT",
  "status": "STATUS_SUCCESS",
  "status_code": "0x0",
  "session_id": 4398046511121,
  "tree_id": 1,
  "share": "\\\\admin-pc\\c$",
  "share_type": "FILE"
}
```


따라서, 이 로그는 클라이언트가 `\\admin-pc\c$` 공유 폴더에 성공적으로 연결되었음을 기록한 것입니다. 이 연결이 성공하면 클라이언트는 해당 공유 폴더에 저장된 파일들에 접근할 수 있게 됩니다.

## DCERPC fields

```
"smb": {
  "id": 4,
  "dialect": "unknown",
  "command": "SMB2_COMMAND_IOCTL",
  "status": "STATUS_SUCCESS",
  "status_code": "0x0",
  "session_id": 4398046511201,
  "tree_id": 0,
  "dcerpc": {
    "request": "REQUEST",
    "response": "RESPONSE",
    "opnum": 0,
    "req": {
      "frag_cnt": 1,
      "stub_data_size": 136
    },
    "res": {
      "frag_cnt": 1,
      "stub_data_size": 8
    },
    "call_id": 2
  }
}
```


DCERPC(Distributed Computing Environment / Remote Procedure Call) 필드는 네트워크 상에서 원격 프로시저 호출을 분석할 때 중요한 정보를 제공합니다. DCERPC는 클라이언트가 원격 서버에서 함수를 실행할 수 있도록 하는 프로토콜이며, 주로 Windows 환경에서 사용됩니다. 이 프로토콜의 필드를 이해하는 것은 보안 분석, 네트워크 모니터링, 그리고 문제 해결에서 매우 유용합니다..

## DCERPC BINDACK

- 설명:
  - **BINDACK** 메시지는 서버가 클라이언트의 BIND 요청을 처리한 후 보내는 응답입니다. 서버가 클라이언트의 요청을 승인하면, **BINDACK** 메시지를 보내고, 이후 클라이언트는 해당 인터페이스에 대한 RPC 호출을 시작할 수 있습니다.
  - **BINDACK** 메시지에는 서버가 요청을 승인했음을 알리는 정보와 함께, 해당 인터페이스와의 통신에 사용할 전송 구문(transfer syntax) 등이 포함됩니다.


```
"dcerpc": {
  "request": "BIND",
  "response": "BINDACK",
  "interfaces": [
    {
      "uuid": "12345778-1234-abcd-ef00-0123456789ac",
      "version": "1.0",
      "ack_result": 2,
      "ack_reason": 0
    },
    {
      "uuid": "12345778-1234-abcd-ef00-0123456789ac",
      "version": "1.0",
      "ack_result": 0,
      "ack_reason": 0
    },
    {
      "uuid": "12345778-1234-abcd-ef00-0123456789ac",
      "version": "1.0",
      "ack_result": 3,
      "ack_reason": 0
```


이 로그는 클라이언트가 여러 인터페이스에 대해 BIND 요청을 보내고, 서버가 각 인터페이스에 대해 다른 결과를 반환한 것을 보여줍니다:

- 첫 번째 인터페이스(UUID: 12345778-1234-abcd-ef00-0123456789ac, 버전: 1.0)에 대해서는 서버가 접근을 거부했습니다 (`ack_result: 2`).
- 두 번째 인터페이스(UUID 동일, 버전 동일)에 대해서는 접근이 허용되었습니다 (`ack_result: 0`).
- 세 번째 인터페이스(UUID 동일, 버전 동일)에 대해서는 이미 바인딩이 존재하거나 중복된 요청으로 간주하여 처리되었습니다 (`ack_result: 3`).

## NTLMSSP fields

이 예시는 NTLMSSP 인증 과정에서 사용되는 필드와 SMB(서버 메시지 블록) 프로토콜의 세션 설정 과정에서 해당 필드들이 어떻게 사용되는지를 보여줍니다. NTLMSSP는 Windows 환경에서 사용되는 인증 프로토콜로, 사용자와 도메인 정보를 통해 인증을 처리합니다.


```
"ntlmssp": {
  "domain": "VNET3",
  "user": "administrator",
  "host": "BLU",
  "version": "60.230 build 13699 rev 188"
}
```

```
"smb": {
  "id": 3,
  "dialect": "NT LM 0.12",
  "command": "SMB1_COMMAND_SESSION_SETUP_ANDX",
  "status": "STATUS_SUCCESS",
  "status_code": "0x0",
  "session_id": 2048,
  "tree_id": 0,
  "ntlmssp": {
    "domain": "VNET3",
    "user": "administrator",
    "host": "BLU",
    "version": "60.230 build 13699 rev 188"
  },
  "request": {
    "native_os": "Unix",
    "native_lm": "Samba 3.9.0-SVN-build-11572"
  },
  "response": {
    "native_os": "Windows (TM) Code Name \"Longhorn\" Ultimate 5231",
    "native_lm": "Windows (TM) Code Name \"Longhorn\" Ultimate 6.0"
  }
}
```

## Kerberos fields

Kerberos는 네트워크 보안 프로토콜로, 특히 대규모 네트워크 환경에서 사용자와 서비스 간의 안전한 인증을 제공합니다. 이 프로토콜은 사용자가 네트워크 상의 서비스에 접근할 때 신원을 확인하고, 세션 키를 사용해 통신 내용을 암호화합니다.


```
"smb": {
  "dialect": "2.10",
  "command": "SMB2_COMMAND_SESSION_SETUP",
  "status": "STATUS_SUCCESS",
  "status_code": "0x0",
  "session_id": 35184439197745,
  "tree_id": 0,
  "kerberos": {
    "realm": "CONTOSO.LOCAL",
    "snames": [
      "cifs",
      "DC1.contoso.local"
    ]
  }
}
```


# Event type: BITTORRENT-DHT

비트토렌트 **DHT**는 여러 컴퓨터들이 서로 파일을 주고받을 때, 그 파일이 어디에 있는지 찾는 데 도움을 주는 시스템이에요. 여기서 **"**노드**"**라는 건 각각의 컴퓨터를 뜻해요. 컴퓨터들이 서로 **"**여기 있어요**!"**라고 알려주면서, 파일을 찾을 수 있게 도와주는 거죠.

```
"bittorrent_dht": {
  "transaction_id": "0c17",
  "client_version": "4c540126",
  "request_type": "ping",
  "request": {
    "id": "41aff1580119f074e2f537f231f12adf684f0d1f"
  }
}

"bittorrent_dht": {
  "transaction_id": "0c17",
  "client_version": "5554b50c",
  "response": {
    "id": "42aeb304a0845b3b9ee089327b48967b8e87b2e2"
  }
}
```


이 두 개의 로그는 컴퓨터 A가 "거기 있어요?"라고 다른 컴퓨터 B에게 물어봤고, 컴퓨터 B가 "네, 여기 있어요!"라고 대답한 과정을 보여줘요. 이걸 통해 컴퓨터들이 서로 잘 연결되어 있는지 확인할 수 있는 거예요.


```
"bittorrent_dht": {
  "transaction_id": "420f0000",
  "client_version": "5554b50c",
  "request_type": "find_node",
  "request": {
    "id": "37579bad1bad166af4329508096fae8c553c6cf4",
    "target": "37579bad1bad166af4329508096fae8c553c6cf4"
  }
}
```


### **find_node** 요청이 뭐예요**?**

**find_node**는 말 그대로 "어떤 컴퓨터를 찾아줘!"라고 요청하는 거예요. BitTorrent DHT 시스템에서, 컴퓨터 A가 네트워크 상에서 다른 컴퓨터 B를 찾고 싶을 때, "find_node" 요청을 보내요. 이 요청을 받으면, 네트워크는 그 컴퓨터가 어디 있는지 또는 그 컴퓨터에 가까운 다른 컴퓨터들의 위치를 알려줍니다.

# Event type: SSH


```
"ssh": {
  "client": {
      "proto_version": "2.0",
      "software_version": "OpenSSH_6.7",
      "hassh": {
          "hash": "ec7378c1a92f5a8dde7e8b7a1ddf33d1",
          "string": "curve25519-sha256,diffie-hellman-group14-sha256,diffie-hellman-group14-sha1,ext-inf
      }
   },
  "server": {
      "proto_version": "2.0",
      "software_version": "OpenSSH_6.7",
      "hassh": {
          "hash": "ec7378c1a92f5a8dde7e8b7a1ddf33d1",
          "string": "curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256",
      }
   }
}
```


보안 관점에서 Suricata에서 이 SSH 객체를 분석할 때 주의 깊게 살펴야 할 필드는 다음과 같습니다:

1. 소프트웨어 버전 (**`software_version`**):
    - 구버전 소프트웨어의 사용은 보안 취약점을 초래할 수 있습니다. 최신 버전으로 업데이트하여 보안성을 강화해야 합니다.
2. **Hassh** 문자열 (**`hassh.string`**):
    - 사용된 암호화 알고리즘을 평가하여, SHA-1과 같은 취약한 알고리즘이 사용되지 않도록 주의해야 합니다. 더 안전한 SHA-256 기반 알고리즘을 선호해야 합니다.
3. **Hassh** 해시 (**`hassh.hash`**):
    - MD5 해시의 사용으로 인해 보안상의 위험이 있을 수 있으므로, 이 해시가 잘못된 동작이나 비정상적인 트래픽을 탐지하는 데 유효한지 평가해야 합니다.

# Event type: Flow


```
"flow": {
  "pkts_toserver": 23,
  "pkts_toclient": 21,
  "bytes_toserver": 4884,
  "bytes_toclient": 7392,
  "bypassed": {
    "pkts_toserver": 10,
    "pkts_toclient": 8,
    "bytes_toserver": 1305,
    "bytes_toclient": 984
  },
  "start": "2019-05-28T23:32:29.025256+0200",
  "end": "2019-05-28T23:35:28.071281+0200",
  "age": 179,
  "bypass": "capture",
  "state": "bypassed",
  "reason": "timeout",
  "alerted": false
}
```


Suricata에서 이 `flow` 객체를 분석할 때, 보안 관점에서 주의 깊게 살펴야 할 주요 필드는 다음과 같습니다:

1. **`state`: "bypassed"`** - 트래픽이 보안 검사를 우회했음을 나타내며, 잠재적인 위협 탐지의 누락 가능성을 시사합니다.
2. **`reason`: "timeout"`** - 시간 초과로 인해 트래픽이 우회된 이유를 나타내며, 성능 문제나 잘못된 구성으로 인한 것인지 확인이 필요합니다.
3. **`alerted`: false** - 우회된 트래픽에서 보안 경고가 발생하지 않았음을 나타내며, 경고 규칙의 적절한 작동 여부를 검토해야 합니다.
4. **`bypassed`** 객체 - 우회된 트래픽의 양과 패킷 수를 나타내며, 많은 트래픽이 우회된 경우 보안 리스크를 증가시킬 수 있습니다.
5. **`start`** 및 **`end`** - 트래픽 흐름의 시작과 종료 시간을 분석하여, 비정상적인 활동 여부를 확인할 수 있습니다.

6. **bypass: "capture"`** - 특정 트래픽이 캡처되지 않고 우회된 이유를 나타내며, 이 설정이 의도된 것인지 검토가 필요합니다.

이러한 필드들을 집중적으로 분석함으로써, 네트워크 보안 상태를 평가하고, 잠재적인 위험 요소를 식별할 수 있습니다. 특히 우회된 트래픽이 왜 발생했는지, 그리고 이로 인해 어떤 보안 허점이 생겼는지를 이해하는 것이 중요합니다.

## Event type: RDP .

RDP 이벤트 로그는 원격 데스크톱 연결에서 발생하는 초기 협상 과정을 기록하며, 보안 관점에서 매우 중요한 정보를 제공합니다. 이러한 로그를 통해 RDP 연결의 보안 상태를 평가하고, 잠재적인 보안 위협을 조기에 탐지할 수 있습니다. 특히, RDP는 공격자가 자주 노리는 대상이므로, 이러한 로그를 통해 네트워크를 철저히 모니터링하고 보안 조치를 강화하는 것이 중요합니다.

### 보안 관점에서 주의해야 할 사항

1. 프로토콜 및 암호화 수준 평가: 사용된 RDP 버전과 암호화 방법이 충분히 강력한지 확인해야 합니다. 구버전의 RDP나 약한 암호화 방법은 보안 위험을 증가시킬 수 있습니다.
2. 비정상적인 연결 시도 탐지: 자주 반복되는 연결 시도, 무단 접근 시도 등 비정상적인 패턴이 있는지 로그를 통해 분석해야 합니다.
3. 로그 모니터링: RDP 로그를 지속적으로 모니터링하여, 의심스러운 활동이 발견될 경우 신속하게 대응할 수 있도록 해야 합니다.

```
"rdp": {
  "tx_id": 0,
  "event_type": "initial_request",
  "cookie": "A70067"
}

"rdp": {
  "tx_id": 1,
  "event_type": "initial_response"
}

"rdp": {
  "tx_id": 2,
  "event_type": "connect_request",
  "client": {
    "version": "v5",
    "desktop_width": 1152,
    "desktop_height": 864,
    "color_depth": 15,
    "keyboard_layout": "en-US",
    "build": "Windows XP",
    "client_name": "ISD2-KM84178",
    "keyboard_type": "enhanced",
    "function_keys": 12,
    "product_id": 1,
    "capabilities": [
      "support_errinfo_pdf"
    ],
    "id": "55274-OEM-0011903-00107"
  },
  "channels": [
    "rdpdr",
    "cliprdr",
    "rdpsnd"
  ]
}

"rdp": {
  "tx_id": 3,
  "event_type": "connect_response"
}
```


Suricata에서 이 RDP 객체를 분석할 때, 보안 관점에서 주의 깊게 살펴야 할 필드는 다음과 같습니다:

1. `event_type`: RDP 세션의 진행 상태를 나타내며, 비정상적인 연결 시도를 감지하는 데 중요합니다.
2. `cookie`: 세션 식별을 위한 쿠키 값으로, 세션 하이재킹 시도를 탐지하는 데 사용될 수 있습니다.

3. `client`: 클라이언트의 운영 체제, RDP 버전, 이름, 기능 등 중요한 보안 정보를 제공합니다. 특히, 구버전 소프트웨어나 보안 지원이 종료된 운영 체제 사용은 위험할 수 있습니다.
4. `channels`: 클라이언트가 요청하는 RDP 채널로, 이들 채널을 통해 발생할 수 있는 보안 위험을 평가해야 합니다.
5. `tx_id`: 이벤트의 순서를 추적하여, 예상치 못한 연결 흐름이나 비정상적인 활동을 탐지할 수 있습니다.

이러한 필드를 집중적으로 분석함으로써 RDP 세션의 보안 상태를 평가하고, 잠재적인 위험 요소를 식별할 수 있습니다. 특히, 운영 체제의 버전, 클라이언트 기능, 요청된 채널에 주의하여 네트워크 보안을 강화하는 것이 중요합니다.

# Event type: RFB

RFB 이벤트는 VNC 프로토콜을 사용하는 원격 접속 활동을 모니터링하는 데 중요한 역할을 합니다. Suricata에서 RFB 이벤트를 분석할 때, 보안 담당자는 클라이언트와 서버의 버전 정보, 사용된 보안 유형, 인증 결과 및 화면 업데이트와 같은 필드를 주의 깊게 살펴야 합니다. 이러한 정보를 통해 비인가된 원격 접속 시도를 조기에 탐지하고, 필요 시 즉각적인 대응을 수행할 수 있습니다. VNC와 같은 원격 접속 도구는 강력한 보안 설정과 주의 깊은 모니터링이 필요하므로, 이러한 이벤트에 대한 지속적인 감시가 중요합니다.


```
"rfb": {
  "server_protocol_version": {
    "major": "003",
    "minor": "007"
  },
  "client_protocol_version": {
    "major": "003",
    "minor": "007"
  },
  "authentication": {
    "security_type": 2,
    "vnc": {
      "challenge": "0805b790b58e967f2b350a0c99de3881",
      "response": "aecb26faeaaa62179636a5934bac1078"
    },
    "security-result": "OK"
  },
  "screen_shared": false,
  "framebuffer": {
    "width": 1280,
    "height": 800,
    "name": "foobar@localhost.localdomain",
    "pixel_format": {
      "bits_per_pixel": 32,
      "depth": 24,
      "big_endian": false,
      "true_color": true,
      "red_max": 255,
      "green_max": 255,
      "blue_max": 255,
      "red_shift": 16,
      "green_shift": 8,
      "blue_shift": 0
    }
  }
```

Suricata에서 이 RFB 객체를 분석할 때, 보안 관점에서 주의 깊게 살펴야 할 필드는 다음과 같습니다:

1. 프로토콜 버전 (**`server_protocol_version`**, **`client_protocol_version`**): 구버전의 RFB 프로토콜 사용으로 인한 보안 취약점을 평가해야 합니다.
2. 인증 (**`authentication`**): VNC 암호 인증 방식과 인증 과정에서 발생한 데이터를 주의 깊게 살펴보고, 가능한 추가적인 암호화 방안을 고려해야 합니다.
3. 화면 공유 (**`screen_shared`**): 화면 공유가 비활성화되어 있는지 확인하고, 활성화된 경우 이를 통해 발생할 수 있는 정보 유출 위험을 평가해야 합니다.
4. 프레임버퍼 설정 (**`framebuffer`**): 원격 화면의 이름과 해상도, 픽셀 형식 등이 예상치 못한 정보 노출을 유발할 수 있는지 확인해야 합니다.
5. 보안 유형 (**`security_type`**): VNC의 기본 암호 인증이 사용된 경우, 이는 보안상의 위험을 내포할 수 있으며, 더 강력한 보안 조치가 필요할 수 있습니다.

이러한 요소들을 분석함으로써, 원격 접속 세션의 보안 상태를 평가하고, 잠재적인 위험 요소를 조기에 탐지하여 대응할 수 있습니다. VNC와 같은 원격 접속 도구는 특히 민감한 데이터에 접근할 때 강력한 보안 설정이 필요합니다.

# Event type: MQTT

EVE-JSON 출력 형식에서 **MQTT**(Message Queuing Telemetry Transport) 프로토콜에 대한 출력은 MQTT 트랜잭션마다 하나의 객체로 구성됩니다. 이 객체에는 공통 필드와 다양한 유형별 필드가 포함됩니다. Suricata의 EVE-JSON 출력에서 MQTT 트랜잭션을 분석할 때, 이러한 필드들은 특정 MQTT 메시지에 대한 정보를 제공하며, 보안 및 네트워크 관리에서 중요한 역할을 합니다.

(MQTT(Message Queuing Telemetry Transport)라는 것은 인터넷을 통해 서로 정보를 주고받을 때 사용하는 특별한 방법이에요. 예를 들어, 스마트 전구가 자신의 상태를 스마트폰에게 알려줄 때 사용될 수 있어요. Suricata라는 프로그램은 이 정보를 잘 관찰하고 기록해서, 나쁜 일이 일어나는 걸 막기 위해 노력해요.

이 프로그램은 MQTT로 주고받는 메시지 하나하나를 기록하는데, 이 기록을 **EVE-JSON**이라는 특별한 형식으로 저장해요. 이렇게 저장된 기록 하나를 객체라고 부르는데, 이 객체는 MQTT 메시지에 대한 중요한 정보를 담고 있어요. 이 정보들은 메시지가 어디서 왔고 어디로 가는지, 무슨 내용인지 등을 알려줘요.)

```
{
  "timestamp": "2020-05-19T18:00:39.016985+0200",
  "flow_id": 1454127794305760,
  "pcap_cnt": 65,
  "event_type": "mqtt",
  "src_ip": "0000:0000:0000:0000:0000:0000:0000:0001",
  "src_port": 60105,
  "dest_ip": "0000:0000:0000:0000:0000:0000:0000:0001",
  "dest_port": 1883,
  "proto": "TCP",
  "mqtt": {
    "publish": {
      "qos": 2,
      "retain": false,
      "dup": false,
      "topic": "house/bulbs/bulb1",
      "message_id": 3,
      "message": "OFF"
    },
    "pubrec": {
      "qos": 0,
      "retain": false,
      "dup": false,
      "message_id": 3
    },
    "pubrel": {
      "qos": 1,
      "retain": false,
      "dup": false,
      "message_id": 3
    },
    "pubcomp": {
      "qos": 0,
      "retain": false,
      "dup": false,
      "message_id": 3
    }
  }
}
```


### 1. `src_ip` 및 `dest_ip`

- 설명: 메시지의 출발지와 목적지 IP 주소.
- 보안 중요성: 비정상적인 외부 접근을 확인.

### 2. `dest_port`: 1883

- 설명: 메시지가 전송된 포트 번호.
- 보안 중요성: 비표준 포트 사용 시 보안 취약 가능성.

### 3. `mqtt.publish`

- 설명: MQTT 메시지 게시 정보 (`topic`, `message` 등).

- 보안 중요성: 민감한 주제나 메시지가 노출될 위험.

4. `message_id`: 3

- 설명: 메시지의 고유 식별자.
- 보안 중요성: 메시지 추적 및 중복 여부 확인.

5. `pubrec`, `pubrel`, `pubcomp`

- 설명: QoS 2 처리 상태 메시지.
- 보안 중요성: 정상적인 메시지 전달 과정 확인.

6. `dup`: false

- 설명: 메시지가 중복되었는지 여부.
- 보안 중요성: 중복된 메시지로 인한 네트워크 부담 방지.

7. `retain`: false

- 설명: 메시지가 서버에 저장되는지 여부.
- 보안 중요성: 민감한 데이터가 의도하지 않게 저장되지 않도록 확인.

## Event type: HTTP2


```
"http2": {
  "request": {
    "settings": [
      {
        "settings_id": "SETTINGSMAXCONCURRENTSTREAMS",
        "settings_value": 100
      },
      {
        "settings_id": "SETTINGSINITIALWINDOWSIZE",
        "settings_value": 65535
      }
    ]
  },
  "response": {}
}
```

**SETTINGSMAXCONCURRENTSTREAMS (**최대 동시 스트림 수**)**:

- 설명: 한 번에 서버와 클라이언트가 동시에 처리할 수 있는 요청의 최대 개수를 설정합니다.
- 보안 관점: 너무 많은 스트림을 허용하면 서버가 과부하에 걸려 서비스 거부 공격(DDoS)에 취약할 수 있습니다. 적절한 값을 설정하면 이런 공격을 방지할 수 있습니다.

**SETTINGSINITIALWINDOWSIZE (**초기 윈도우 크기**)**:

- 설명: 클라이언트가 서버로부터 한 번에 받을 수 있는 데이터의 양을 설정합니다.
- 보안 관점: 이 값을 조정해 데이터 전송을 조절함으로써 네트워크 혼잡이나 악의적인 대량 데이터 전송을 방지할 수 있습니다.

```
"request": {
  "headers": [
    {
      "name": ":authority",
      "value": "localhost:3000"
    },
    {
      "name": ":method",
      "value": "GET"
    },
    {
      "name": ":path",
      "value": "/doc/manual/html/index.html"
    },
    {
      "name": ":scheme",
      "value": "http"
    },
    {
      "name": "accept",
      "value": "*/*"
    },
    {
      "name": "accept-encoding",
      "value": "gzip, deflate"
    },
    {
      "name": "user-agent",
      "value": "nghttp2/0.5.2-DEV"
    }
  ]
},
"response": {
  "headers": [
    {
      "name": ":status",
      "value": "200"
    },
    {
      "name": "server",
      "value": "nghttpd nghttp2/0.5.2-DEV"
    },
    {
      "name": "content-length",
      "value": "22617"
    },
    {
      "name": "cache-control",
      "value": "max-age=3600"
    },
    {
      "name": "date",
      "value": "Sat, 02 Aug 2014 10:50:25 GMT"
    },
    {
      "name": "last-modified",
      "value": "Sat, 02 Aug 2014 07:58:59 GMT"
```

# Event type: PGSQL

GSQL 이벤트 로그는 데이터베이스와 클라이언트 간의 대화 기록입니다. 보안 관점에서 중요한 포인트는 다음과 같습니다:

1. 트랜잭션 추적: 로그를 통해 누가 어떤 데이터를 요청하고, 어떤 응답을 받았는지 추적할 수 있습니다. 이를 통해 비정상적인 접근이나 의심스러운 활동을 감지할 수 있습니다.
2. 데이터 유출 방지: 로그에 저장된 쿼리와 응답을 통해 민감한 정보가 외부로 유출되지 않도록 확인할 수 있습니다.
3. 접근 통제: 어떤 IP와 포트가 데이터베이스에 접근했는지 확인하여, 허가되지 않은 접근을 방지하고 차단할 수 있습니다.

**request**: 클라이언트에서 보낸 SELECT 쿼리.
**response**: 서버에서 보낸 응답, 여기에는 필드 설명(RowDescription), 실제 데이터(DataRow), 그리고 쿼리 완료 메시지(CommandComplete)가 포함됩니다.


```
{
  "timestamp": "2021-11-24T16:56:24.403417+0000",
  "flow_id": 1960113262002448,
  "pcap_cnt": 780,
  "event_type": "pgsql",
  "src_ip": "172.18.0.1",
  "src_port": 54408,
  "dest_ip": "172.18.0.2",
  "dest_port": 5432,
  "proto": "TCP",
  "pgsql": {
    "tx_id": 4,
    "request": {
      "simple_query": "select * from rule limit 5000;"
    },
    "response": {
      "field_count": 7,
      "data_rows": 5000,
      "data_size": 3035751,
      "command_completed": "SELECT 5000"
    }
  }
}
```

# Event type: IKE

`IKE` 이벤트는 VPN 연결을 설정할 때 발생하는 보안 프로토콜 관련 로그입니다. IKE에는 두 가지 버전이 있는데, `version_major` 필드로 IKEv1(1)과 IKEv2(2)를 구분할 수 있습니다. 각 버전은 `ikev1` 또는 `ikev2`라는 별도의 객체에 고유한 정보를 담고 있습니다.

보안 관점에서 중요한 점:

- 버전 구분: IKEv1과 IKEv2는 보안 기능과 동작이 다릅니다. 로그에서 버전을 구분하면, 각 버전의 특징에 맞게 보안 분석을 할 수 있습니다.
- 고유 정보: 각 버전에 특화된 보안 설정과 정보를 포함하고 있어, 이를 통해 연결 설정이 안전하게 이루어졌는지 확인할 수 있습니다.

이 로그는 VPN 설정과 관련된 보안 이벤트를 모니터링하고, 잠재적인 보안 위협을 식별하는 데 유용합니다.