“확장 가능한 이벤트 형식(Extensible Event Format, 줄여서 **EVE**) 이벤트 로그는 **JSON** 형식으로 기록됩니다.”

Output types

EVE는 여러 가지 출력 방법을 지원할 수 있습니다. `regular`는 일반 파일로 출력을 의미하며, 이외에도 다양한 출력 옵션이 있습니다. 예를 들어, `syslog`는 시스템 로그로 출력을 보내는 것이며, `unix_dgram`과 `unix_stream`은 각각 Unix 도메인 데이터그램 소켓과 스트림 소켓으로 출력을 보내는 방식입니다. `redis`는 Redis 데이터베이스로 로그를 전송하는 옵션입니다. 이러한 다양한 출력 방법을 사용하면 로그를 원하는 대로 관리하고 저장할 수 있습니다.

Alerts

경고(Alerts)는 규칙에 일치한 이벤트 기록입니다. 이러한 경고는 메타데이터와 함께 추가될 수 있으며, 예를 들어 경고가 생성된 애플리케이션 계층 기록(HTTP, DNS 등) 및 규칙의 요소들이 포함될 수 있습니다. 이는 경고에 대한 추가적인 맥락 정보를 제공하여, 분석 시 어떤 상황에서 경고가 발생했는지를 더 잘 이해할 수 있게 합니다.

```
alert:
  #payload: yes             # enable dumping payload in Base64
  #payload-buffer-size: 4kb # max size of payload buffer to output in eve-log
  #payload-printable: yes   # enable dumping payload in printable (lossy) format
  #payload-length: yes      # enable dumping payload length, including the gaps
  #packet: yes              # enable dumping of packet (without stream segments)
  #http-body: yes           # Requires metadata; enable dumping of http body in Base64
  #http-body-printable: yes # Requires metadata; enable dumping of http body in printabl

  # metadata:

    # Include the decoded application layer (ie. http, dns)
    #app-layer: true

    # Log the current state of the flow record.
    #flow: true

    #rule:
      # Log the metadata field from the rule in a structured
      # format.
      #metadata: true

      # Log the raw rule text.
      #raw: false
```

Suricata의 `alert` 섹션에서 사용 가능한 다양한 옵션들을 설명하고 있습니다. 각 항목은 주석으로 비활성화되어 있지만, 활성화할 경우 Suricata가 이벤트 로그에 포함할 내용을 제어합니다. 각 옵션의 의미는 다음과 같습니다:

1. **payload**: Base64 형식으로 페이로드(데이터의 본문)를 덤프할지를 결정합니다. 이 옵션을 활성화하면 경고가 발생할 때 패킷의 페이로드가 로그에 기록됩니다.
2. **payload-buffer-size**: 페이로드 버퍼의 최대 크기를 설정합니다. `eve-log`에 출력될 페이로드 버퍼의 최대 크기를 4KB로 설정합니다.
3. **payload-printable**: 페이로드를 출력 가능한 형식으로 덤프할지를 결정합니다. 이는 손실 가능성이 있는 포맷으로 페이로드를 출력합니다.
4. **payload-length**: 페이로드의 길이를 덤프할지를 결정합니다. 갭을 포함한 페이로드 길이가 로그에 기록됩니다.
5. **packet**: 패킷(스트림 세그먼트를 제외한)을 덤프할지를 결정합니다. 이 옵션을 활성화하면 패킷이 로그에 기록됩니다.
6. **http-body**: HTTP 본문을 Base64 형식으로 덤프할지를 결정합니다. 메타데이터가 필요합니다.
7. **http-body-printable**: HTTP 본문을 출력 가능한 형식으로 덤프할지를 결정합니다. 메타데이터가 필요합니다.
8. **metadata**: 이 섹션은 경고에 포함할 메타데이터 옵션들을 정의합니다.
    - **app-layer**: 디코딩된 애플리케이션 계층 데이터(예: HTTP, DNS)를 로그에 포함할지를 결정합니다.
    - **flow**: 현재 흐름(flow) 기록의 상태를 로그에 포함할지를 결정합니다.
    - **rule**: 규칙의 메타데이터 필드를 구조화된 형식으로 로그에 기록할지를 결정합니다.

이 설정을 통해 Suricata가 생성하는 경고 로그에 얼마나 많은 정보가 포함될지를 세밀하게 조정할 수 있습니다. 이 옵션들은 보안 분석가가 필요로 하는 정보의 양과 깊이를 조절하는 데 사용됩니다.

Anomaly

Suricata의 비정상적인 이벤트 기록에 대한 설명입니다. 이 이벤트는 예상치 못한 값이 포함된 패킷이 처리될 때 생성됩니다. 이러한 이벤트는 다음과 같은 조건을 포함합니다:

- 잘못된 프로토콜 값: 패킷의 프로토콜 값이 올바르지 않은 경우.
- 잘못된 프로토콜 길이 값: 프로토콜 길이 값이 올바르지 않은 경우.
- 기타 의심스러운 조건: 패킷을 의심스럽게 만드는 기타 조건들.

또한, 스트림의 정상적인 진행 중에도 발생할 수 있는 조건이 있는데, 이는 스트림 이벤트라고 하며, 올바르지 않은 값이 포함된 제어 시퀀스나 예상 순서에서 벗어난 제어 시퀀스를 포함합니다.

이러한 비정상적인 이벤트는 유형에 따라 보고되고 구성됩니다:

1. **Decode**: 패킷을 해석(디코드)하는 과정에서 발생하는 비정상적인 이벤트.
2. **Stream**: 스트림 진행 중 발생하는 비정상적인 이벤트.
3. **Application layer**: 애플리케이션 계층에서 발생하는 비정상적인 이벤트.

"anomaly" 설정은 네트워크에서 발생할 수 있는 예상치 못한 상황이나 비정상적인 동작을 기록하는 데 사용됩니다. 여기서 비정상적인 동작은 패킷이 손상되었거나 예상하지 못한 값이 포함된 경우 등입니다.

주요 내용:

- 비정상 로그는 손상된 패킷이나 잘못된 IP/UDP/TCP 길이 값을 가진 패킷 등 추가 처리에 적합하지 않은 패킷을 기록하며, 이미 설정된 스트림에서 예상하지 못한 동작을 설명합니다. 네트워크에서 비정상적인 상황이 자주 발생하면 패킷 처리 성능이 저하될 수 있습니다.
- 보고되는 비정상 사항:
    - **Decode**: 개별 패킷을 디코딩하는 동안 감지된 잘못된 값과 조건. 여기에는 낮은 레벨 프로토콜 길이에 대한 잘못된 값이나 예상하지 못한 값도 포함됩니다.
    - **Stream**: 스트림과 관련된 이벤트(TCP 3-way 핸드셰이크 문제, 예상하지 못한 시퀀스 번호 등)를 포함합니다.

    - **Application layer**: 애플리케이션 계층에서 발생한 예상하지 못한 또는 잘못된 조건을 나타냅니다.
- 기본적으로, 비정상 로그 기록은 비활성화되어 있으며, 비정상 로그 기록을 활성화하면 애플리케이션 계층의 비정상 보고도 활성화됩니다.
- 로그 기록 선택:
    - `types` 설정에서 `decode`, `stream`, `applayer` 중 하나 또는 모두를 선택할 수 있습니다.
    - `packethdr` 설정을 통해 패킷 비정상에 대한 패킷 헤더의 로그 기록 여부를 결정할 수 있습니다.

설정 예시:

- `types`에서 `decode`, `stream`, `applayer` 설정을 사용하여 특정 유형의 비정상 로그를 활성화하거나 비활성화할 수 있습니다.
- `packethdr` 옵션을 사용하여 패킷 비정상에 대한 패킷 헤더 로그 기록을 활성화할 수 있습니다.

이 설정은 네트워크 보안을 강화하기 위해 비정상적인 네트워크 동작을 모니터링하고 로그로 기록할 수 있도록 도와줍니다.

HTTP

```
- http:
    extended: yes     # enable this for extended logging information
    # custom allows additional http fields to be included in eve-log
    # the example below adds three additional fields when uncommented
    #custom: [Accept-Encoding, Accept-Language, Authorization]
    # set this value to one among {both, request, response} to dump all
    # http headers for every http request and/or response
    # dump-all-headers: [both, request, response]
```

Suricata에서 HTTP 트랜잭션을 로그로 기록하는 방법을 구성하는 설정입니다. 이를 통해 HTTP 요청 및 응답과 관련된 다양한 정보를 기록할 수 있습니다.

주요 설정 항목:

- **extended: yes**

    - 이 옵션을 `yes`로 설정하면, HTTP 트랜잭션에 대한 확장된(더 자세한) 로그 정보를 활성화할 수 있습니다. 기본 로그 정보 외에도 더 많은 HTTP 관련 데이터가 기록됩니다.
- **custom**
    - 이 옵션을 사용하면 추가적으로 포함하고 싶은 HTTP 필드를 정의할 수 있습니다. 예를 들어, `Accept-Encoding`, `Accept-Language`, `Authorization` 같은 필드를 추가적으로 로그에 포함시킬 수 있습니다.
    - 주석이 제거되면(앞의 `#` 기호를 제거하면), 이 필드들이 로그에 기록됩니다.

예시: .yaml
코드 복사: `custom: [Accept-Encoding, Accept-Language, Authorization]`

- **dump-all-headers**
    - 이 옵션은 HTTP 요청 및/또는 응답에 대한 모든 HTTP 헤더를 로그로 기록할지 여부를 설정합니다.
    - 가능한 값:
        - `both`: 요청과 응답의 모든 헤더를 로그로 기록합니다.
        - `request`: 요청의 모든 헤더만 로그로 기록합니다.
        - `response`: 응답의 모든 헤더만 로그로 기록합니다.

예시: .yaml
코드 복사: `dump-all-headers: both`

예시: .yaml
코드 복사:

```
http:
  extended: yes
  custom: [Accept-Encoding, Accept-Language, Authorization]
  dump-all-headers: both
```

이 설정은 HTTP 트랜잭션과 관련된 다양한 정보를 수집하고 분석할 수 있도록 도와줍니다. 특히, 웹 애플리케이션의 동작이나 보안 관련 사항을 모니터링하고자 할 때 유용합니다.

DNS

"DNS 레코드는 요청과 응답 각각에 대해 하나씩 로그로 기록됩니다."

Suricata의 EVE 로그에서 DNS 이벤트가 기록되는 방식을 의미합니다. 즉, DNS 쿼리(요청)와 DNS 응답은 별도의 로그 항목으로 기록되어, 각 DNS 트랜잭션에 대해 두 개의 로그 엔트리가

생성된다는 것입니다. 이를 통해 요청과 응답의 세부 정보를 개별적으로 확인하고 분석할 수 있습니다.

예를 들어, 어떤 도메인에 대한 DNS 쿼리가 발생하면, 그 쿼리와 관련된 요청 로그가 기록되고, 이후 서버로부터 받은 응답에 대한 로그도 별도로 기록됩니다. 이는 DNS 활동을 추적하고 분석할 때 매우 유용합니다.

```
dns:
  #version: 3

  # Enable/disable this logger. Default: enabled.
  #enabled: yes

  # 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: no
  #responses: no

  # Format of answer logging:
  # - detailed: array item per answer
  # - grouped: answers aggregated by type
  # Default: all
  #formats: [detailed, grouped]

  # Types to log, based on the query type.
  # Default: all.
  #types: [a, aaaa, cname, mx, ns, ptr, txt]
```

이 설정은 Suricata의 DNS 로그에 대한 설정 부분을 설명하는 것입니다. 각 항목의 기능을 살펴보겠습니다.

- `version: 3`: 이 부분은 DNS 로그 형식의 버전을 지정하는 옵션입니다. 주석 처리되어 있지만, 필요에 따라 버전을 명시적으로 설정할 수 있습니다.
- `enabled: yes`: 이 설정은 DNS 로깅 기능을 활성화하거나 비활성화하는 옵션입니다. 기본값은 활성화(`yes`)입니다.
- `requests: no`, `responses: no`: 이 옵션들은 DNS 쿼리(요청)와 응답에 대한 로깅을 제어합니다.
    - `requests: no`는 DNS 쿼리 로그를 비활성화하는 옵션입니다.
    - `responses: no`는 DNS 응답 로그를 비활성화하는 옵션입니다.
    - 기본값으로는 요청과 응답 모두가 로그로 기록됩니다.
- `formats: [detailed, grouped]`: DNS 응답의 로그 형식을 제어합니다.
    - `detailed`: 각 응답 항목을 별도의 배열 항목으로 기록합니다.
    - `grouped`: 응답을 유형별로 그룹화하여 기록합니다.
    - 기본값으로는 모든 형식으로 기록됩니다.
- `types: [a, aaaa, cname, mx, ns, ptr, txt]`: 로깅할 DNS 쿼리 유형을 설정합니다.
    - 기본값으로는 모든 유형의 DNS 쿼리가 로그로 기록됩니다.
    - 명시적으로 특정 유형의 DNS 쿼리만 기록하고 싶다면, 여기에 해당 유형을 나열할 수 있습니다 (예: `A`, `AAAA`, `CNAME`, `MX` 등).

이 설정을 통해 DNS 요청 및 응답에 대한 로그 기록을 세부적으로 제어할 수 있습니다. 필요에 따라 쿼리 또는 응답만 기록하거나, 특정 유형의 DNS 쿼리만 기록할 수 있습니다.

TLS

"TLS 기록은 각 세션당 하나의 기록으로 로그됩니다."

이 의미는 TLS(Transport Layer Security) 통신이 이루어질 때, 하나의 세션에 대해 단일 로그 항목이 생성된다는 것입니다. 즉, 각 TLS 연결에 대해 하나의 로그 레코드가 기록되며, 이 로그에는 해당 세션의 시작부터 종료까지의 정보가 포함됩니다.

이 방식은 세션 단위로 TLS 통신의 보안 정보를 추적하는 데 유용하며, 각 로그 항목에는 주로 인증서 정보, 암호화 버전, 세션 시작 및 종료 시간 등이 포함될 수 있습니다.

.yaml 에서…

```
- tls:
    extended: yes     # 확장된 로깅 정보를 활성화합니다.
    # custom allows to control which tls fields that are included
    # in eve-log
    #custom: [subject, issuer, serial, fingerprint, sni, version, not_before, not_after, c
```

- **extended: yes**
  - 확장된 로깅 정보를 활성화합니다. 이 설정을 통해 TLS 세션에 대한 더 많은 세부 정보가 로그에 포함됩니다.
- **custom**
  - `custom` 옵션은 EVE 로그에 포함할 TLS 필드를 선택적으로 지정할 수 있게 합니다. 이를 통해 로그의 크기를 줄이거나, 필요한 정보만 포함되도록 커스터마이징할 수 있습니다.
  - 예를 들어, 주석에 나와 있는 `custom: [subject, issuer, serial, fingerprint, sni, version, not_before, not_after, certificate, chain, ja3, ja3s, ja4]` 설정은 아래의 필드들을 포함하도록 지정합니다:
    - **subject**: TLS 인증서의 주체(일반적으로 도메인 이름).
    - **issuer**: 인증서를 발급한 기관.
    - **serial**: 인증서의 고유 일련 번호.
    - **fingerprint**: 인증서의 해시 값, 주로 SHA-256 해시.
    - **sni**: Server Name Indication, 클라이언트가 요청하는 도메인 이름.
    - **version**: 사용된 TLS 프로토콜 버전 (예: TLS 1.2, TLS 1.3).
    - **not_before**: 인증서가 유효하기 시작한 날짜.
    - **not_after**: 인증서가 만료되는 날짜.

    - **certificate**: 전체 인증서 데이터.
    - **chain**: 인증서 체인 데이터.
    - **ja3**: 클라이언트 핑거프린팅 해시 값.
    - **ja3s**: 서버 핑거프린팅 해시 값.
    - **ja4**: 추가적으로 사용할 수 있는 핑거프린팅 데이터.

이 설정을 통해 TLS 로그에 어떤 정보가 포함될지 매우 세밀하게 제어할 수 있습니다. 예를 들어, 특정 상황에서 보안 분석에 필요한 특정 필드만 기록할 수 있습니다.

ARP

"ARP(주소 결정 프로토콜) 로그는 하나의 요청(ARP Request)과 하나의 응답(ARP Response)으로 각각 하나의 기록으로 로그됩니다."

설명:

- **ARP Request**: 네트워크에서 특정 IP 주소에 해당하는 MAC 주소를 찾기 위해 보내는 요청입니다. 네트워크 상의 모든 기기에 이 요청이 전달되며, 해당 IP 주소를 가진 기기는 자신의 MAC 주소를 응답으로 보내게 됩니다.
- **ARP Response**: ARP 요청에 대한 응답으로, 요청한 IP 주소에 해당하는 MAC 주소를 보내는 것입니다. 이 응답은 요청을 보낸 기기에만 전달됩니다.

로그 기록:

Suricata는 이러한 ARP 요청과 응답을 각각 하나의 기록으로 로그에 남깁니다. 이를 통해 네트워크 상에서 발생하는 ARP 트래픽을 모니터링할 수 있으며, ARP 스푸핑 공격과 같은 네트워크 보안 위협을 감지하는 데 유용하게 사용될 수 있습니다.

이 방식으로 로그를 남김으로써 네트워크 트래픽의 변동 사항을 추적하고, 네트워크 상의 장치 간의 통신이 올바르게 이루어지고 있는지를 확인할 수 있습니다.

YAML:

```
- arp:
    enabled: no
```

로그는 기본적으로 비활성화되어 있습니다. 이는 ARP(주소 결정 프로토콜)가 매우 많은 이벤트를 생성할 수 있기 때문

Date modifiers in filename

“Suricata의 EVE-JSON 로그 출력 설정에서 파일 이름에 날짜 수정자를 사용할 수 있습니다.”

```
outputs:
  - eve-log:
      filename: eve-%s.json
```

위의 예에서는 파일 이름에 에포크 시간을 추가합니다. `%s`는 현재 시간을 초 단위로 나타내는 C 라이브러리의 날짜 수정자입니다.

모든 C 라이브러리의 날짜 수정자가 지원됩니다. 파일 이름에 다양한 날짜 형식을 추가할 수 있으며, 사용 가능한 모든 수정자는 `strftime`의 매뉴얼 페이지에서 확인할 수 있습니다.

이 기능을 통해 로그 파일의 이름에 날짜나 시간을 포함시켜, 파일 관리와 아카이빙을 보다 용이하게 할 수 있습니다. 예를 들어, 로그 파일을 날짜별로 나누어 저장하거나, 특정 시간대의 로그만을 쉽게 식별할 수 있게 됩니다.

Rotate log file

“Eve-log는 시간을 기준으로 로그 파일을 자동으로 교체(회전)하도록 설정할 수 있습니다.”

예를 들어, 매일 저녁 12시에 Suricata가 새로운 로그 파일을 만들도록 설정할 수 있어요. 이렇게 하면 매일매일의 기록이 다른 파일에 저장되기 때문에, 이전 날의 기록과 헷갈리지 않게 돼요. 새로운 하루가 시작되면 Suricata는 새로운 파일에 기록을 시작하게 되죠.

요약

- 로그 파일 회전: 정해진 시간마다 새로운 로그 파일을 만드는 기능
- 장점: 기록을 날짜별로 구분할 수 있어서 관리하기 쉬워짐

```
outputs:
  - eve-log:
      filename: eve-%Y-%m-%d-%H:%M.json
      rotate-interval: minute
```

위의 예시는 매 분마다 새로운 로그 파일을 생성하며, 파일 이름에는 타임스탬프(시간 정보)가 포함됩니다. 이 외에도 로그 파일을 교체하는 기준으로 시간(시간 단위)이나 하루(일 단위)를 사용할 수 있습니다.

또한, 로그 파일 교체 간격을 상대적인 값으로 지정할 수도 있습니다. 예를 들어, X초마다 로그 파일을 교체하도록 설정할 수 있습니다.

요약

- 타임스탬프 포함: 파일 이름에 시간 정보를 포함하여 매 분마다 새로운 로그 파일을 생성.
- 지원되는 간격: 분, 시간, 하루 단위로 로그 파일을 교체할 수 있음.
- 상대적인 간격 설정: 예를 들어, X초마다 로그 파일을 교체하도록 설정 가능.

Multiple Logger Instances

EVE' 로거 인스턴스를 여러 개 설정할 수 있습니다. 이는 각 인스턴스가 서로 다른 설정을 사용하여 로그를 기록할 수 있음을 의미합니다. 예를 들어, 하나의 EVE 인스턴스는 기본적인 로그를 기록하고, 다른 인스턴스는 특정 이벤트만을 기록하도록 설정할 수 있습니다.

```
outputs:
  - eve-log:
      enabled: yes
      type: file
      filename: eve-ips.json
      types:
        - alert
        - drop

  - eve-log:
      enabled: yes
      type: file
      filename: eve-nsm.json
      types:
        - http
        - dns
        - tls
```

- 위의 예에서는 "alerts"와 "drops" 로그가 'eve-ips.json' 파일에 기록되며, "http", "dns", 그리고 "tls" 로그는 'eve-nsm.json' 파일에 기록됩니다.
- 여기서 중요한 점은 **drop** 유형의 로거는 단 한 번만 사용할 수 있다는 것입니다. 즉, **drop** 유형의 로거는 여러 번 정의할 수 없으며, 한 번만 설정할 수 있습니다. 반면에, **alert**, **http**, **dns**, **tls** 등 다른 유형의 로거는 필요에 따라 여러 번 정의할 수 있습니다.
- 이러한 설정은 로그를 관리하고 분석하는 데 유용하며, 서로 다른 로그 데이터를 별도의 파일에 기록하여 보다 체계적인 로그 관리가 가능합니다. 다만, **drop** 로거는 단일 인스턴스로만 설정 가능하다는 점을 유의해야 합니다.
- 정할 수 있습니다.
