

수업 내용 정리 보고서

작성일: 2026년 7월 8일

챕터 구성

- Suricata IDS/NSM 엔진의 구조와 운영 실습
- Port-Security를 이용한 L2 네트워크 접근 제어
- 계층별 보안 메커니즘의 한계와 통합 분석 체계

Suricata IDS/NSM 엔진의 구조와 운영 실습

네트워크 보안 관제와 분석의 핵심은 단순히 공격을 탐지하는 것을 넘어, 패킷이 유입되는 시점부터 로그가 생성되기까지의 전체 흐름을 이해하는 것입니다. 본 장에서는 오픈소스 기반의 고성능 IDS/IPS/NSM 엔진인 Suricata의 개념부터 설치, 내부 패킷 처리 파이프라인, 그리고 Snort와의 비교 실습을 통해 실무적인 운영 능력을 학습합니다.

1. Suricata의 개념 및 역할

Suricata는 네트워크 트래픽을 실시간으로 분석하여 알려진 공격 패턴을 탐지하는 IDS(Intrusion Detection System, 침입 탐지 시스템)이자, 인라인 구조에서 직접 차단까지 수행하는 IPS(Intrusion Prevention System, 침입 방지 시스템) 역할을 수행합니다.

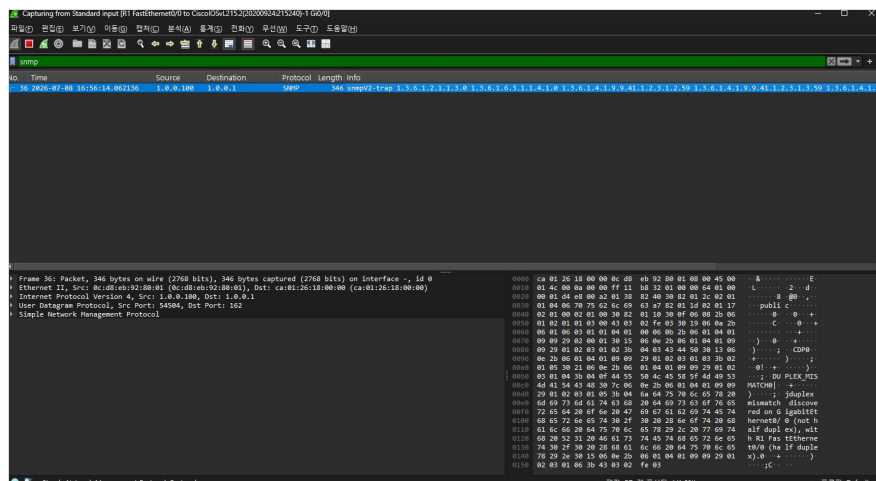
- **NSM(Network Security Monitoring):** 단순한 탐지를 넘어, 누가 누구와 통신했는지(Flow), 어떤 프로토콜을 사용했는지(HTTP, DNS, TLS 등)와 같은 메타 데이터를 수집하여 네트워크 전반의 가시성을 확보하는 활동입니다.
- **SIEM(Security Information and Event Management) 연동:** Suricata가 생성한 EVE JSON 형식의 로그는 Filebeat, Logstash 등을 통해 SIEM으로 전송되어, 방화벽, 웹 서버 로그 등과 결합하여 하나의 공격 흐름으로 분석하는 상관분석의 기초 자료가 됩니다.

특히 SIEM 연동 시, SNMP Trap과 같은 네트워크 관리 프로토콜을 통해 장비의 상태나 특정 이벤트를 수집하여 통합 관제 체계를 구축합니다.

실습 화면 해석

네트워크 장비에서 발생하는 특정 이벤트(Trap)가 패킷 형태로 전송될 때, Wireshark를 통해 UDP 162번 포트로 전송되는 SNMPv2-trap 패킷의 상세 페이로드를 확인할 수 있습니다. 이는 IDS가 수집하는 다양한 이벤트 로그의 기초가 되는 패킷 레벨의 데이터입니다.

Wireshark의 snmpV2-trap 필터링 결과 및 소스/목적지 IP 정보



Wireshark를 통해 확인한 SNMPv2-trap 프로토콜의 상세 페이로드 및 패킷 구조

이와 같이 구조화된 패킷 데이터가 Suricata의 엔진을 거치면 eve.json과 같은 분석 가능한 로그 형태로 변환됩니다.

2. Suricata 설치 및 환경 설정

Suricata의 설치 과정은 단순한 프로그램 설치가 아니라, 어떤 라이브러리를 포함해 빌드하느냐에 따라 지원 기능(AF_PACKET, Hyperscan, Lua 등)이 결정되는 과정입니다.

2.1 빌드 및 설치 핵심 절차

소스 코드를 다운로드한 후 ./configure 단계에서 설치 경로(--prefix)와 설정 파일 경로(--sysconfdir)를 지정합니다. 이후 make와 make install을 통해 컴파일 및 설치를 진행하며, suricata-update를 통해 최신 탐지 룰셋을 적용합니다.

설치 후 suricata --build-info 명령어를 통해 AF_PACKET 지원 여부, Hyperscan 라이브러리 적용 여부 등을 확인하여 고성능 캡처 환경이 준비되었는지 검증해야 합니다.

2.2 주요 설정 파일(suricata.yaml) 최적화

운영 환경에 맞는 설정은 탐지 효율과 시스템 성능에 직결됩니다.

- **HOME_NET:** 보호 대상 네트워크 대역을 지정합니다. (예: [172.16.14.98]) 이를 정확히 설정해야 룰셋의 \$HOME_NET 변수가 정상 작동합니다.
- **af-packet:** 패킷 캡처 인터페이스(예: ens160)를 지정합니다. 인터페이스 이름이 일치하지 않으면 패킷 수신에 불가능합니다.
- **rule-files:** 실습 시에는 local.rules만 활성화하여 작성한 룰의 동작 여부를 빠르게 확인하는 것이 효율적입니다.
- **eve-log:** 로그 저장 공간과 성능 관리를 위해 필요한 프로토콜(HTTP, DNS, TLS 등)만 선택적으로 활성화합니다.

3. 하드웨어 및 커널 레벨의 패킷 수신 경로

Suricata가 패킷을 분석하기 전, 패킷은 하드웨어와 커널의 복잡한 경로를 거쳐 사용자 공간(User Space)으로 전달됩니다.

3.1 CPU와 멀티스레드 구조

Suricata는 멀티스레드 기반 엔진입니다. 하나의 프로세스 내에서 여러 스레드가 패킷 수신, 분석, 탐지, 로그 처리를 분담합니다. 이를 위해 CPU의 논리 코어 수에 맞춰 스레드를 배정하는 detect-thread-ratio 설정 등이 사용됩니다.

3.2 패킷의 이동 경로 (Data Path)

- 1. **NIC & RX Ring**: 패킷이 NIC에 도착하면 RX Ring 버퍼에 저장됩니다. 이때 **RSS(Receive Side Scaling)**를 통해 여러 RX Queue로 분산하여 CPU 부하를 나눕니다.
- 2. **DMA (Direct Memory Access)**: CPU를 거치지 않고 NIC가 직접 메모리 버퍼로 패킷 데이터를 전송하여 CPU 부하를 최소화합니다.
- 3. **NIC Driver & NAPI**: 드라이버는 sk_buff(skb)라는 커널 자료구조를 생성합니다. 효율적인 수신을 위해 인터럽트(Interrupt) 방식과 폴링(Polling) 방식을 결합한 **NAPI** 구조를 사용합니다.
- 4. **Kernel Network Stack**: L2(Ethernet) → L3(IP) → L4(TCP/UDP) 계층을 거치며 헤더가 해석됩니다.
- 5. **Socket Buffer & Application**: 최종적으로 소켓 버퍼에 저장된 데이터를 Suricata와 같은 애플리케이션이 읽어 들입니다.

4. Suricata 내부 패킷 처리 파이프라인

Suricata 엔진 내부에서는 패킷이 다음과 같은 단계적 파이프라인을 통해 처리됩니다.

단계	설명 및 핵심 기능	주의사항 및 포인트
Capture	AF_PACKET, DPDK 등을 통해 패킷을 수집	캡처 방식에 따라 커널 우회 여부 결정
Decode	L2~L4 헤더 해석 및 터널링(VXLAN 등) 해제	정확한 디코딩이 되어야 이후 분석 가능
Defrag	IP 조각화(Fragmentation) 패킷 재조립	host-os-policy 설정으로 OS별 재조립 방식 일치 필요
Flow Tracking	5-tuple 기반으로 통신 흐름을 묶어 flow_id 부여	동일 세션의 패킷을 하나로 관리하는 기준
Stream/Reassembly	TCP 세그먼트를 재조립하여 Payload 복원	depth 설정값 초과 시 재조립 불가 (탐지 우회 가능성)
App-Layer Parser	HTTP, DNS, TLS 등 상위 프로토콜 필드 추출	event_type: http 로그가 생성되는 단계
Detect Engine	룰셋과 비교하여 공격 패턴 매칭 (MPM, Hyperscan 사용)	가장 자원 소모가 많은 단계이며 최적화가 필수적
Output	fast.log, eve.json, pcap-log 출력	SIEM 연동의 최종 결과물 생성

5. Snort와 Suricata 비교 및 실습 분석

두 엔진 모두 룰셋 호환성이 높지만, 아키텍처와 로그 출력 방식에서 큰 차이가 있습니다.

- **처리 방식**: Snort(전통적 단일 스레드 중심) vs Suricata(멀티스레드 기반 고성능 처리).
- **로그 관점**: Snort는 Alert 중심의 로그를 남기는 반면, Suricata는 **NSM 관점**에서 Flow, HTTP, DNS 등 상세 메타데이터를 JSON 형태로 남깁니다.

flow_id를 이용한 상관분석 실습

실습을 통해 동일한 룰을 적용했을 때, Snort의 alert_json은 단순 매칭 결과만을 보여주지만, Suricata의 eve.json은 flow_id라는 고유 식별자를 제공합니다.

분석가는 flow_id를 기준으로 event_type: alert(탐지), event_type: http(요청/응답 내용), event_type: flow(통신량, 시간) 로그를 모두 연결할 수 있습니다. 이를 통해 "어떤 공격 룰에 걸렸는가"를 넘어 "공격자가 어떤 URL로 어떤 데이터를 보냈고, 서버는 어떻게 응답했는가"라는 전체 컨텍스트를 파악할 수 있습니다.

6. 네트워크 보안 계층의 상호 보완 (Port-Security)

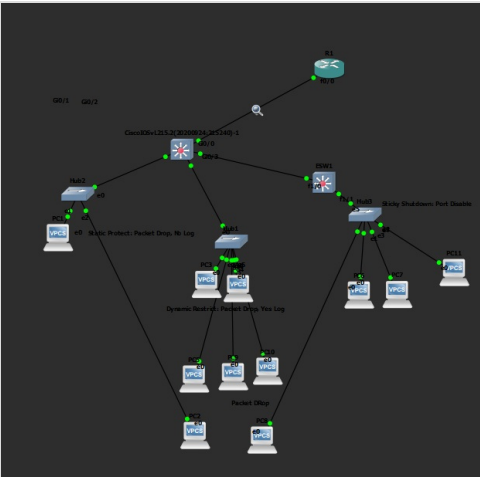
Suricata가 L3~L7 계층의 트래픽 내용을 분석한다면, 스위치의 **Port-Security**는 L2 계층에서 물리적 접속을 제어합니다.

- **Port-Security**: 특정 포트에 허용된 Source MAC 주소의 개수(Maximum)를 제한하거나 특정 MAC만 허용(Sticky/Static)하여 비인가 단말의 접속을 차단합니다. 위반 시 Shutdown 모드를 통해 포트를 err-disabled 상태로 만들어 물리적으로 격리합니다.
- **한계점**: Port-Security는 MAC 주소 기반이므로, 공격자가 허용된 MAC을 복제하는 **MAC Spoofing**이나, IP-MAC 매핑 정보를 속이는 **ARP Spoofing** 공격을 원천적으로 막을 수 없습니다.

실습 화면 해석

다양한 보안 모드(Static Protect, Dynamic Restrict, Sticky Shutdown)의 동작 차이를 검증하기 위해 설계된 토폴로지입니다. 각 허브와 연결된 PC들이 스위치 포트의 보안 설정에 따라 패킷이 드롭(Drop)되거나 포트가 비활성화(Disable)되는 과정을 확인합니다.

도식 내 Sticky Shutdown, Static Protect, Dynamic Restrict 등 각 영역별 설정 라벨



Shutdown, Restrict, Protect 모드별 동작 차이를 실습하기 위해 설계된 네트워크 토폴로지

L2 수준의 Port-Security로 1차 진입을 제어하고, 이를 통과한 트래픽을 Suricata IDS가 L7 수준에서 정밀 분석하는 것이 계층적 방어(Defense in Depth)의 핵심입니다.

7. 운영 점검 체크리스트

Suricata를 안정적으로 운영하기 위해 다음 항목을 주기적으로 점검해야 합니다.

- **리소스 모니터링:** 멀티스레드 동작 시 CPU 코어별 부하가 균등한지, memcap 설정으로 인해 메모리 부족 현상이 없는지 확인합니다.
- **패킷 드롭 확인:** NIC의 RX Ring Buffer가 가득 차 패킷 드롭이 발생하는지 `ethtool -S` 등으로 점검합니다.
- **물셋 최적화:** 불필요한 물은 주석 처리하고, `suricata-update`를 통해 최신 시그니처를 유지합니다.
- **로그 무결성:** `eve.json` 파일이 정상적으로 생성되고 있으며, SIEM으로 누락 없이 전송되고 있는지 확인합니다.

Port-Security를 이용한 L2 네트워크 접근 제어

네트워크의 가장 하단인 Layer 2(데이터 링크 계층)에서는 스위치가 MAC 주소를 학습하여 프레임을 전달한다. 하지만 기본 설정의 스위치는 어떤 장비가 연결되더라도 MAC 주소를 자동으로 학습하고 통신을 허용하기 때문에, 인가되지 않은 노트북이나 허브를 무단으로 연결하는 보안 취약점이 존재한다. **포트 보안(Port-Security)**은 이러한 취약점을 해결하기 위해 특정 스위치 포트에 접속할 수 있는 MAC 주소의 개수나 특정 주소를 제한하는 접근 제어 기능이다.

1. Port-Security의 핵심 동작 원리

포트 보안의 핵심은 **Source MAC 주소(출발지 MAC 주소)**를 검사하는 것이다. 스위치는 포트에 들어오는 프레임의 목적지(Destination)가 아닌, 프레임을 보낸 장비의 주소를 확인하여 허용된 목록(Secure MAC Address)에 있는지 판단한다.

- **검사 기준:** 포트에 유입되는 프레임의 Source MAC 주소
- **판단 프로세스:** 포트 보안 활성화 → 유입 프레임의 Source MAC 확인 → 허용 목록 및 최대 개수(Maximum) 비교 → 허용 또는 위반(Violation) 처리
- **주요 목적:** 비인가 단말의 네트워크 접근 차단, 하나의 포트에 여러 장비(허브/소형 스위치)를 연결하는 행위 방지, MAC Flooding 공격 완화

2. MAC 주소 등록 방식 비교

스위치가 어떤 MAC 주소를 '허용된 주소'로 인식하게 할 것인지에 따라 세 가지 등록 방식을 사용한다.

등록 방식	특징	설정 및 저장 방식	장단점
Static(정적)	관리자가 수동으로 MAC 주소를 지정	설정 파일에 직접 입력하여 저장	보안성이 가장 높으나, 장비가 많을 경우 관리 부담이 매우 큼
Sticky(스티키)	처음 연결된 장비의 MAC을 자동 학습	학습된 주소를 running-config에 자동 기록	편의성과 보안성을 모두 잡은 방식으로, 실무에서 가장 많이 사용됨
Dynamic(동적)	자동 학습하지만 저장하지 않음	메모리 상에만 유지 (재부팅 시 삭제)	설정이 간편하지만 지속적인 보안 유지력이 낮음

특히 Sticky MAC 방식의 경우, 학습된 주소가 `running-config`에 기록되므로, 장비를 재부팅한 후에도 설정을 유지하려면 반드시 `copy running-config startup-config` 또는 `write(wr)` 명령어를 통해 `startup-config`(부팅 설정 파일)에 저장해야 한다.

3. 위반 처리 모드(Violation Mode) 분석

허용되지 않은 MAC 주소가 유입되거나, 설정된 최대 개수(Maximum)를 초과했을 때 스위치가 취하는 대응 방식이다.

- **Protect:** 위반 프레임을 조용히 폐기(Drop)한다. 로그를 남기지 않고 위반 카운터도 증가시키지 않으며, 포트 상태는 그대로 유지한다.

- **Restrict:** 위반 프레임을 폐기하고, SNMP Trap이나 Syslog를 통해 **위반 사실을 기록**한다. 위반 카운터가 증가하며 관리자가 상황을 인지할 수 있다.
- **Shutdown:** 가장 강력한 조치로, 위반 즉시 포트를 **err-disabled** 상태로 전환하여 완전히 차단한다. 포트 LED가 빨간색으로 변하며, 관리자의 수동 복구가 필요하다.

4. 실습 화면 해석: 모드별 적용 사례

실습에서는 각 위반 모드(Protect, Restrict, Shutdown)의 동작 차이를 명확히 확인하기 위해 서로 다른 설정이 적용된 토폴로지를 구성하였다.

- **PC1 구간:** Static MAC과 Protect 모드를 적용하여, 인가되지 않은 패킷은 드롭되지만 로그가 남지 않는 환경을 구축함.
- **PC3, PC4 구간:** Dynamic MAC과 Restrict 모드를 적용하여, 위반 시 패킷은 차단하되 관리자 로그가 생성되는 환경을 구축함.
- **PC11 구간:** Sticky MAC과 Shutdown 모드를 적용하여, 비인가 장비 연결 시 포트가 즉시 비활성화되는 환경을 구축함.

위 토폴로지를 통해 확인할 수 있듯이, 동일한 L2 접근 제어 목적이라도 운영 환경(단순 차단, 모니터링 필요, 즉시 격리 등)에 따라 적절한 Violation Mode를 선택하는 것이 중요하다.

5. 포트 상태 관리 및 복구 (err-disabled)

Shutdown 모드에서 위반이 발생하면 포트는 **err-disabled** 상태가 된다. 이 상태에서는 `no shutdown`을 입력해도 즉시 복구되지 않으며, 포트를 완전히 내렸다가 다시 올려야 한다.

```
conf t
interface gil/0
shutdown      # 포트를 완전히 종료
no shutdown   # 포트를 다시 활성화하여 err-disabled 해제
end
```

매번 수동으로 복구하는 번거로움을 줄이기 위해 **자동 복구(errdisable recovery)** 설정을 사용할 수 있다. 이는 특정 시간이 지나면 스위치가 자동으로 포트를 다시 살려보는 기능이다.

```
conf t
errdisable recovery cause psecure-violation # 포트 보안 위반으로 인한 차단 건을 복구 대상으로 지정
errdisable recovery interval 300           # 300초 후 자동 복구 시도
end
```

단, 자동 복구는 포트를 다시 '활성화'하는 것이지 '보안 정책을 해제'하는 것이 아니다. 위반을 일으킨 장비가 계속 연결되어 있다면, 복구 직후 다시 위반이 발생하여 즉시 **err-disabled** 상태로 되돌아간다.

6. 네트워크 관제 연동 및 로그 분석

Restrict 모드나 Shutdown 모드에서 발생하는 위반 사항은 SNMP(Simple Network Management Protocol) 등을 통해 통합 관제 서버로 전송될 수 있다. 이를 통해 관리자는 터미널에 접속하지 않고도 네트워크 전체의 보안 위반 상태를 실시간으로 모니터링한다.

실제로 위반 이벤트가 발생하면 SNMPv2-trap과 같은 형태로 알람 패킷이 생성되며, 이를 Wireshark로 분석하면 어떤 포트에서 어떤 위반이 발생했는지에 대한 상세 페이로드를 확인할 수 있다.

이러한 로그 분석은 단순한 차단을 넘어, 내부 네트워크에서 MAC 주소 변조 시도가 있는지 혹은 비인가 장비가 어디에 연결되었는지를 추적하는 중요한 근거 자료가 된다.

7. Port-Security의 한계와 보안 보완

포트 보안은 강력한 1차 방어선이지만, MAC 주소라는 식별자의 특성상 완벽한 보안을 보장하지는 않는다.

- **MAC Spoofing:** 공격자가 허용된 장비의 MAC 주소를 알아내어 자신의 NIC 주소를 복제하면 스위치는 이를 정상 장비로 인식하여 통신을 허용한다.
- **ARP Spoofing:** ARP(Address Resolution Protocol)는 IP 주소를 MAC 주소로 변환하는 프로토콜이다. 공격자가 "내가 게이트웨이이다"라는 거짓 ARP 응답을 보내 피해자의 트래픽을 가로채는 ARP Spoofing 공격은 Port-Security(Source MAC 검사)만으로는 막을 수 없다.

결론: Port-Security는 '물리적인 포트 접근'을 제한하는 데 효과적이지만, '논리적인 주소 위조' 공격까지 막으려면 **DHCP Snooping, Dynamic ARP Inspection(DAI)**, 혹은 **802.1X 기반의 사용자 인증**과 같은 상위 보안 기술을 병행 적용해야 한다.

[종합 점검 체크리스트]

- **설정 전:** 포트 모드가 Access로 고정되어 있는가? (Trunk/Dynamic 모드에서는 동작이 불안정할 수 있음)
- **등록 방식:** 장비 교체가 잦은 환경인가? (그렇다면 Aging 설정이나 Sticky MAC 고려)
- **위반 대응:** 보안 수준이 매우 높은 구간인가? (그렇다면 Shutdown 모드 적용)
- **유지 보수:** err-disabled 발생 시 복구 절차를 숙지하고 있는가?
- **검증:** `show port-security interface [포트번호]` 명령으로 현재 학습된 Secure MAC 주소와 위반 횟수를 확인하였는가?

계층별 보안 메커니즘의 한계와 통합 분석 체계

네트워크 보안은 단일 솔루션이 아니라 계층별 방어 체계(Defense in Depth)를 구축하는 과정입니다. 하지만 각 계층의 보안 메커니즘은 고유한 한계를 가지고 있으며, 이를 극복하기 위해서는 하위 계층의 접근 제어와 상위 계층의 통합 분석 체계가 유기적으로 연결되어야 합니다. 본 챕터에서는 L2 계층의 포트 보안부터 IDS/NSM을 통한 통합 분석 흐름까지 학습합니다.

1. L2 계층 보안: Port-Security와 접근 제어

Port-Security는 스위치의 특정 포트에 접속할 수 있는 MAC 주소의 개수를 제한하거나 특정 MAC 주소만 허용하는 Layer 2 보안 기능입니다. 기본적으로 스위치는 MAC 주소 학습(Learning)을 통해 통신하지만, 이를 그대로 두면 허가되지 않은 장비가 네트워크에 무단 접속할 위험이 있습니다.

Port-Security의 핵심 작동 원리

- **검사 기준:** 프레임의 목적지(Destination)가 아닌 **출발지(Source) MAC 주소**를 기준으로 검사합니다.
- **MAC 등록 방식:**
 - **Static:** 관리자가 수동으로 허용할 MAC 주소를 지정합니다.
 - **Sticky:** 처음 연결된 장비의 MAC 주소를 자동으로 학습하여 설정(running-config)에 저장합니다.
 - **Dynamic:** 자동 학습하지만 메모리에만 유지되며, 재부팅 시 사라집니다.
- **위반 처리 모드(Violation Mode):** 허용되지 않은 MAC 주소가 유입되었을 때의 대응 방식입니다.
 - **Protect:** 위반 프레임을 조용히 폐기하며, 로그를 남기지 않습니다.
 - **Restrict:** 프레임을 폐기하고, 로그 기록 및 위반 카운터를 증가시킵니다.
 - **Shutdown:** 즉시 포트를 **err-disabled** 상태로 전환하여 물리적으로 차단합니다. (가장 강력한 조치)

실습 화면 해석: Port-Security 검증 환경

Port-Security의 모드별 동작 차이(Shutdown, Restrict, Protect)와 Sticky MAC 학습 과정을 검증하기 위해, 서로 다른 보안 정책이 적용된 허브와 스위치, 그리고 다수의 VPCS(Virtual PC Simulator)로 구성된 네트워크를 설계하였습니다.

위 토폴로지서에서 확인할 수 있듯이, 각 포트 그룹마다 다른 위반 모드를 설정하여 패킷 드롭 여부와 로그 발생, 포트 상태 변화를 비교 분석할 수 있습니다. 특히 Sticky Shutdown 설정이 적용된 포트의 경우, 허가되지 않은 MAC 주소 유입 시 즉각적으로 포트가 비활성화되는 과정을 통해 L2 접근 제어의 실효성을 확인합니다.

L2 보안의 결정적 한계

Port-Security는 고정 단말 환경에서 유용하지만, 다음과 같은 공격에는 무력합니다.

- **MAC Spoofing:** 공격자가 허용된 장비의 MAC 주소를 복제하여 위장하는 경우, 스위치는 이를 정상 장비로 인식합니다.
- **ARP Spoofing:** Port-Security는 Source MAC만 검사할 뿐, ARP 패킷 내부의 IP-MAC 매핑 정보가 거짓인지 판단하지 못합니다. 따라서 공격자가 게이트웨이의 IP를 자신의 MAC 주소로 속이는 공격은 막을 수 없습니다.

2. 네트워크 레벨 분석: Suricata를 통한 NSM의 구현

L2 보안의 한계를 극복하기 위해서는 단순히 '누가 접속했는가'를 넘어 '어떤 행위를 하는가'를 분석하는 **NSM(Network Security Monitoring)** 체계가 필요합니다. Suricata는 단순한 IDS(침입 탐지 시스템)를 넘어 네트워크의 모든 흐름을 관찰하는 NSM 엔진의 역할을 수행합니다.

패킷 수신부터 애플리케이션 전달까지의 경로

패킷이 NIC를 통해 들어와 사용자 공간의 프로그램(Suricata 등)에 도달하기까지는 복잡한 커널 경로를 거칩니다.

1. **NIC & RX Ring:** 패킷이 하드웨어에 도착하면 RX Queue(Ring Buffer)에 저장됩니다. 이때 **RSS(Receive Side Scaling)**를 통해 트래픽을 여러 CPU 코어로 분산합니다.
2. **DMA (Direct Memory Access):** CPU의 개입 없이 패킷 데이터를 메모리로 직접 전송하여 CPU 부하를 줄입니다.
3. **NIC Driver & skb:** 드라이버가 패킷을 확인하고 리눅스 커널의 핵심 자료구조인 **skb(socket buffer)**를 생성합니다.
4. **NAPI (New API):** 인터럽트(Interrupt) 방식의 부하를 줄이기 위해, 패킷 감지 시에는 인터럽트로 알리고 이후에는 폴링(Polling) 방식으로 패킷을 긁어오는 하이브리드 방식을 사용합니다.
5. **Kernel Network Stack:** L2(Ethernet) → L3(IP) → L4(TCP/UDP) 계층을 순차적으로 처리합니다.
6. **Socket Buffer & Application:** 최종적으로 소켓 버퍼를 거쳐 Suricata와 같은 애플리케이션이 데이터를 읽어갑니다.

3. 통합 분석 파이프라인과 메타데이터 활용

Suricata는 패킷 하나를 단편적으로 보지 않고, 일련의 **파이프라인**을 통해 문맥(Context)을 분석합니다.

Suricata 내부 처리 흐름

캡처(Capture) → 디코딩(Decode) → 재조립(Defrag/Stream) → 플로우 추적(Flow Tracking) → 앱 계층 분석(App-Layer Parser) → 탐지(Detect Engine) → 출력(Output/Logging)

특히 **Flow Tracking** 단계에서 5-tuple(출발지/목적지 IP, 포트, 프로토콜)을 기준으로 flow_id를 부여함으로써, 쪼개져 들어오는 수많은 패킷을 하나의 '통신 세션'으로 묶어 분석할 수 있게 합니다.

EVE JSON과 SIEM 연동의 가치

단순한 Alert 로그(fast.log)는 "공격이 발생했다"는 사실만 알려주지만, **EVE JSON** 형식의 로그는 HTTP URL, DNS 쿼리, TLS 인증서 정보 등 풍부한 메타데이터를 포함합니다. 이를 SIEM(Security Information and Event Management)과 연동하면 서로 다른 이벤트들을 flow_id로 묶어 공격의 전체 타임라인을 재구성할 수 있습니다.

실습 화면 해석: 메타데이터 기반 패킷 분석

단순한 탐지 룰 매칭을 넘어, 실제 네트워크 상에서 흐르는 프로토콜의 상세 페이로드를 분석하는 과정입니다. Wireshark를 통해 SNMPv2-trap 패킷의 내부 구조를 확인하여, 단순한 연결 여부가 아닌 전송되는 데이터의 구체적인 의미를 분석합니다.

위 화면에서 볼 수 있듯이, SNMP trap 패킷의 상세 필드를 분석함으로써 장비의 상태 변화나 특정 이벤트 발생 여부를 정확히 파악할 수 있습니다. 이는 Suricata의 App-Layer Parser가 수행하는 작업과 동일하며, 이렇게 추출된 메타데이터가 JSON 형태로 저장될 때 비로소 고도화된 상관 분석이 가능해집니다.

4. 계층별 보안 요약 및 통합 점검 리스트

L2의 포트 보안과 L3~L7의 IDS/NSM 분석 체계는 상호 보완적입니다. 통합 보안 관점에서의 점검 포인트는 다음과 같습니다.

점검 대상	핵심 확인 항목	판단 기준 및 주의사항
L2 포트 보안	Violation Mode 및 Sticky 설정	단순 접속 차단 목적이라면 Shutdown, 모니터링이 필요하면 Restrict 설정 확인
L2 한계 대응	ARP/MAC Spoofing 가능성	Port-Security만으로는 부족하며, 상위 계층의 트래픽 분석(IDS) 연동 필수
패킷 수신 최적화	RSS 및 NAPI 동작 여부	고트래픽 환경에서 패킷 드롭 발생 시 RX Ring Buffer 및 CPU Affinity 설정 점검
분석 체계	flow_id 기반 상관 분석 가능 여부	단순 Alert 로그가 아닌 EVE JSON의 메타데이터가 SIEM으로 정상 전송되는지 확인

실무 포인트: 보안 사고 분석 시, L2 로그에서 "허가되지 않은 MAC의 접속 시도"를 확인하고, 동시에 Suricata의 flow_id를 추적하여 "해당 MAC이 어떤 내부 서버로 어떤 페이로드를 보냈는지"를 연결하는 것이 통합 분석의 핵심입니다.