

수업 내용 정리 보고서

작성일: 2026년 6월 30일

챕터 구성

- NIDS/NIPS의 개념 및 네트워크 구성 방식
- Snort 패킷 처리 메커니즘 및 데이터 흐름
- 성능 최적화 기술 및 운영 점검 리스트

NIDS/NIPS의 개념 및 네트워크 구성 방식

네트워크 보안의 핵심은 유입되는 트래픽 내에 숨겨진 공격 패턴(Signature)을 식별하고 이에 대응하는 것입니다. 이를 위해 가장 널리 사용되는 오픈소스 도구 중 하나가 Snort이며, Snort를 기반으로 한 시스템을 어떻게 구성하느냐에 따라 **NIDS(Network Intrusion Detection System, 네트워크 침입 탐지 시스템)**와 **NIPS(Network Intrusion Prevention System, 네트워크 침입 방지 시스템)**로 구분됩니다.

1. NIDS와 NIPS의 개념 및 비교

두 시스템의 가장 결정적인 차이는 공격을 감지한 후 '**패킷의 흐름을 물리적으로 차단할 수 있는가**'에 있습니다. 단순히 위험을 알리는 것(Detection)과 실제 위협을 막는 것(Prevention)의 차이입니다.

구분	NIDS (침입 탐지 시스템)	NIPS (침입 방지 시스템)
핵심 기능	공격 시그니처 기반 탐지 및 기록(Logging)	탐지 및 실시간 차단(Packet Drop)
네트워크 위치	트래픽의 복사본을 받는 위치 (Out-of-path)	트래픽이 통과하는 경로 (In-line)
장점	네트워크 성능에 영향이 없음	공격 시 즉각적인 대응 가능
단점/주의사항	공격 차단 불가능 (사후 대응 위주)	장비 성능 저하 시 전체 네트워크 속도 저하

실무 포인트: 실무 환경에서는 성능과 안정성을 고려하여 구성합니다. NIPS는 공격을 즉시 막을 수 있어 강력하지만, 장비의 처리 속도가 전체 네트워크 대역폭을 따라가지 못할 경우 네트워크 병목 현상(Bottleneck)을 유발할 수 있습니다.

2. Snort의 패킷 처리 프로세스 (Packet Processing Pipeline)

Snort와 같은 시스템이 네트워크 패킷을 수신하여 탐지 결과로 출력하기까지는 하드웨어 계층부터 애플리케이션 계층까지 정교한 단계별 과정을 거칩니다. 각 단계는 데이터의 무결성을 유지하면서 분석 효율을 높이기 위해 설계되었습니다.

- NIC (Network Interface Card) 수신:** 케이블을 통해 들어온 물리적 신호를 NIC가 수신하여 하드웨어 버퍼(Hardware Buffer)에 일시적으로 저장합니다.
- DAQ (Data Acquisition) 단계:** 하드웨어 버퍼에 있는 데이터를 실제 프로그램(Snort)이 접근할 수 있는 메모리 영역으로 끌어올리는(Pull) 과정입니다. 이때 libpcap이나 afpacket과 같은 라이브러리가 사용됩니다.
- Decoder (해독):** 수신된 데이터가 어떤 프로토콜(IP, TCP, UDP 등)을 사용하는지 분석하고 규격에 맞게 해독합니다.
- Preprocessor (전처리):** 분절된 패킷을 재조립(Reassembly)하거나 세션(Session) 관련 처리를 수행합니다. 공격자가 패킷을 쪼개서 보내는 등의 우회 기법을 방지하기 위해 필수적인 단계입니다.
- Detection (탐지):** 해독 및 전처리가 완료된 데이터를 미리 정의된 공격 시그니처(Signature)와 비교하여 침입 여부를 판단하고 결과를 출력합니다.

3. 네트워크 구성 방식: 데이터 수집 및 배치

시스템을 어떤 방식으로 네트워크에 연결하느냐에 따라 관제 방식이 완전히 달라집니다.

A. IDS 구성: 포트 미러링(Port Mirroring) 및 TAP

IDS는 네트워크의 흐름을 방해하지 않고 '지켜보는 것'이 목적입니다. 이를 위해 실제 트래픽의 복사본만을 전달받는 방식을 사용합니다.

- 포트 미러링 (Port Mirroring):** 네트워크 스위치의 기능을 이용하여 특정 포트(A)로 오가는 모든 패킷을 다른 포트(B)로 똑같이 복사하여 보내는 방식입니다.
- TAP (Test Access Point):** 네트워크 라인 사이에 물리적인 장치를 연결하여 패킷을 복제해내는 방식입니다.

주의사항: 이 방식들은 트래픽을 가로채지 않고 복사본만 보기 때문에, 탐지는 가능하지만 공격 패킷을 실제로 떨어뜨리는(Drop) 차단 동작은 불가능합니다.

B. IPS 구성: In-line 배치

IPS는 네트워크의 통로 중간에 직접 위치하여 모든 패킷을 검사합니다. 위험한 패킷이 발견되면 해당 패킷을 Drop 시켜 목적지에 도달하지 못하게 만듭니다. 따라서 장비의 처리 속도가 전체 네트워크 품질과 직결됩니다.

4. 성능 최적화 및 고급 기술

고속 네트워크 환경에서 탐지 효율을 높이기 위해 다음과 같은 기술적 요소가 고려됩니다.

- Raw Data 기반 감지:** 프로토콜을 복잡하게 해석하기 전, 원시 데이터(Raw Data) 단계에서 빠르게 감지하는 것이 실무적인 성능 확보에 유리할 수 있습니다.
- Lua 스크립트 및 Lua JIT:** Snort 등에서 복잡한 탐지 로직을 구현할 때 Lua라는 스크립트 언어를 사용합니다. 특히 **Lua JIT (Just-In-Time)** 컴파일러를 사용하면 필요한 부분만 즉석에서 컴파일하여 실행 속도를 획기적으로 높일 수 있어 고속 패킷 처리에 매우 유리합니다.

[학습 점검 리스트]

- [개념 확인]** IDS와 IPS의 가장 큰 차이점인 'Packet Drop' 가능 여부를 설명할 수 있는가?
- [구성 판단]** 네트워크 성능 저하를 최소화하면서 보안 관제만 수행하고 싶을 때 적합한 구성 방식은 무엇인가? (정답: 포트 미러링을 이용한 IDS)

- [프로세스 이해] NIC에서 Detection 단계까지 패킷이 이동하는 순서를 알고 있는가? (NIC → DAQ → Decoder → Preprocessor → Detection)
- [실무 고려] IPS를 구축할 때 가장 주의해야 할 성능 관련 이슈는 무엇인가? (정답: 장비 처리 속도가 네트워크 전체 속도에 미치는 영향)

Snort 패킷 처리 메커니즘 및 데이터 흐름

네트워크 보안 관제의 핵심은 유입되는 트래픽 내에 숨겨진 공격 시그니처(Signature)를 얼마나 빠르고 정확하게 찾아내느냐에 있습니다. 이번 장에서는 오픈소스 네트워크 침입 탐지 시스템(NIDS)인 **Snort**를 중심으로, 네트워크 패킷이 하드웨어 단계에서부터 최종 탐지 단계에 이르기까지 어떤 과정을 거쳐 처리되는지 그 내부 메커니즘을 상세히 분석합니다.

1. Snort의 정의 및 역할

Snort는 네트워크 트래픽을 실시간으로 분석하여 침입을 탐지하거나 차단하는 오픈소스 기반의 **NIDS(Network Intrusion Detection System, 네트워크 침입 탐지 시스템)**입니다. Snort는 미리 정의된 공격 규칙(Rule/Signature)과 일치하는 패킷이 네트워크를 통과할 때 이를 탐지하고 기록하며, 설정에 따라 해당 패킷의 흐름을 제어하는 역할을 수행합니다.

2. IDS와 IPS: 감지와 차단의 결정적 차이

Snort를 운용할 때 가장 먼저 결정해야 하는 설계 방향은 시스템을 **IDS(탐지 전용)**로 운영할 것인지, 아니면 **IPS(차단 포함)**로 운영할 것인지입니다. 이 둘의 차이는 단순히 기능의 유무를 넘어, 네트워크 물리 구조와 성능에 막대한 영향을 미칩니다.

구분	IDS (Intrusion Detection System)	IPS (Intrusion Prevention System)
핵심 기능	감지(Detection) 및 알람/기록	차단(Prevention/Drop) 및 대응
패킷 처리 방식	복사본을 받아 분석 (Passive)	실제 트래픽 경로에 위치 (Inline)
네트워크 연결	포트 미러링(Port Mirroring) 또는 TAP 이용	네트워크 스위치/방화벽 사이의 직접 연결
성능 영향	전체 네트워크 속도에 영향을 주지 않음	Snort 처리 속도가 전체 네트워크 속도를 결정

[실무 포인트]

- IDS 운영 시:** 트래픽의 복사본만 가져오기 때문에 공격을 발견하더라도 이미 패킷은 목적지에 도달한 상태입니다. 따라서 '차단'보다는 '사후 분석'과 '관제'에 목적을 둡니다.
- IPS 운영 시:** 위험 패킷을 실제로 *Drop(폐기)*할 수 있어 즉각적인 방어가 가능합니다. 하지만 Snort의 처리 과정에서 지연(Latency)이 발생하면 전체 네트워크 성능이 저하될 수 있으므로, 고성능 처리가 필수적입니다.

3. Snort 패킷 처리 5단계 프로세스

패킷이 네트워크 선로를 타고 들어와 Snort에 의해 탐지되기까지는 하드웨어 계층부터 애플리케이션 계층까지 이어지는 정밀한 데이터 흐름이 존재합니다.

- NIC (Network Interface Card) 단계:** 케이블을 통해 들어온 전기적/광학적 신호를 네트워크 카드가 수신합니다. 수신된 데이터는 가장 먼저 하드웨어 자체의 **하드웨어 버퍼(Hardware Buffer)**에 임시 저장됩니다.
- DAQ (Data Acquisition) 단계:** 하드웨어 버퍼에 있는 데이터를 실제 Snort 프로그램이 접근할 수 있는 **운영체제의 메모리 영역**으로 퍼 올리는 단계입니다.
 - 이 단계에서는 *libpcap*이나 *afpacket*과 같은 라이브러리가 활용되어 효율적인 데이터 수집을 담당합니다.
- Decoder (해독) 단계:** 수집된 Raw Data를 프로토콜 규격(Ethernet, IP, TCP/UDP 등)에 따라 분석하여 유의미한 패킷 구조로 해독합니다.
- Preprocessor (전처리) 단계:** 해독된 패킷들을 바탕으로 데이터를 재조립(Reassembly)하거나 세션(Session) 관련 처리를 수행합니다. 파편화된 패킷을 하나로 모으거나, 공격자가 탐지를 피하기 위해 사용하는 데이터 조작을 방어하는 중요한 단계입니다.
- Detection (탐지) 단계:** 전처리가 완료된 데이터를 미리 등록된 **공격 시그니처(Signature)**와 대조합니다. 일치하는 패턴이 발견되면 로그를 남기거나(IDS), 패킷을 차단(IPS)하는 결과를 출력합니다.

4. 네트워크 구현 및 성능 최적화 기술

포트 미러링(Port Mirroring)을 통한 관제

IDS를 구축할 때 가장 흔히 사용하는 방식은 스위치 허브의 **포트 미러링** 기능을 사용하는 것입니다. 이는 특정 포트(A)로 오가는 모든 트래픽을 복사하여 다른 포트(B)로 똑같이 보내주는 기술입니다. 이를 통해 네트워크 트래픽 흐름을 방해(Interruption)하지 않으면서도 트래픽의 복사본을 안전하게 가로채어 관제할 수 있습니다. 다만, 복사본을 보는 것이므로 실제 공격 패킷을 차단할 수는 없습니다.

Lua 및 Lua JIT를 활용한 성능 향상

Snort의 처리 속도와 유연성을 높이기 위해 스크립트 언어인 **Lua**가 사용됩니다. 특히 **Lua JIT(Just-In-Time)** 컴파일러를 활용하면, 실행 시점에 필요한 부분만 미리 컴파일하여 처리하기 때문에 매우 빠른 속도로 스크립트를 실행할 수 있습니다. 이는 대규모 트래픽 환경에서 IPS의 성능 저하 문제를 완화하는 데 도움을 줍니다.

[주의사항] IPS를 구축할 때는 반드시 Snort의 처리 용량이 네트워크 대역폭을 수용할 수 있는지 검증해야 합니다. Snort의 연산 부하가 네트워크 병목 현상(Bottleneck)을 유발하여 서비스 장애로 이어질 수 있기 때문입니다.

학습 확인 질문

- IDS와 IPS를 결정할 때, 네트워크 성능(Latency) 관점에서 고려해야 할 핵심 요소는 무엇인가?
- 패킷이 NIC에 도착한 후, 메모리로 이동하기 위해 거치는 단계는 무엇인가?
- 포트 미러링 방식을 사용했을 때, 네트워크 보안 관제 측면에서의 장점과 한계점은 무엇인가?
- 패킷의 재조립(Reassembly)이 이루어지는 단계는 프로세스 중 어디에 해당하는가?

성능 최적화 기술 및 운영 점검 리스트

네트워크 침입 탐지 시스템(NIDS, Network Intrusion Detection System)인 Snort를 실무에 적용할 때 가장 중요한 결정 사항은 "탐지(Detection)만 할 것인가, 아니면 차단(Blocking)까지 할 것인가"입니다. 이 결정은 네트워크의 구성 방식, 성능 부하, 그리고 보안 정책에 따라 달라지며, 시스템의 성능 최적화 방향을 결정 짓는 핵심 요소가 됩니다.

1. 네트워크 배치 모델: IDS와 IPS의 비교

시스템을 네트워크 상에 어떻게 배치하느냐에 따라 IDS(침입 탐지 시스템)와 IPS(침입 방지 시스템)로 구분됩니다. 두 모델은 패킷을 처리하는 방식과 네트워크에 미치는 영향이 근본적으로 다릅니다.

구분	IDS (Intrusion Detection System)	IPS (Intrusion Prevention System)
핵심 기능	감지 및 기록 (Monitoring)	감지 및 실시간 차단 (Blocking/Packet Drop)
네트워크 구성	측면 배치 (Out-of-path)	직렬 배치 (In-line)
연결 방식	포트 미러링(Port Mirroring) 또는 TAP	네트워크 트래픽 흐름 상에 직접 위치
성능 영향	트래픽 흐름에 영향을 주지 않음	탐지 속도가 전체 네트워크 속도를 결정

[운영 포인트]

- 포트 미러링(Port Mirroring) 활용:** 스위치 허브의 기능을 이용하여 특정 포트(A)로 오는 패킷의 복사본을 다른 포트(B)로 보내는 방식입니다. 이 방식은 실제 트래픽을 가로채지 않고 관제만 하므로 네트워크 지연을 유발하지 않지만, 이미 통과한 패킷을 사후에 확인하는 수준이므로 실시간 차단은 불가능합니다.
- IPS의 병목 현상 주의:** IPS는 위험한 패킷을 실제로 Drop(패킷 폐기)해야 하므로, 패킷이 시스템을 통과하는 시간이 전체 네트워크의 지연 시간(Latency)이 됩니다. 따라서 Snort의 처리 속도가 네트워크의 전체 성능을 좌우하게 됩니다.

2. Snort 패킷 처리 파이프라인 (Packet Processing Pipeline)

Snort가 네트워크 패킷을 수신하여 탐지 결과를 내놓기까지는 하드웨어 계층부터 애플리케이션 계층까지 정해진 단계를 거칩니다. 각 단계의 효율성은 시스템 전체의 처리량(Throughput)과 직결됩니다.

- NIC (Network Interface Card):** 케이블을 통해 들어온 전기적 신호를 수신하여 디지털 데이터로 변환한 후, 하드웨어 버퍼(Buffer)에 저장합니다.
- DAQ (Data Acquisition):** 하드웨어 버퍼에 머물러 있는 데이터를 실제 목적지인 프로그램(Snort)의 메모리 영역으로 퍼 올리는(Fetch) 역할을 합니다. 실무에서는 libpcap이나 afpacket과 같은 라이브러리를 사용하여 이 과정을 최적화합니다.
- Decoder (해독):** 수신된 로우 데이터(Raw data)를 프로토콜(TCP/IP, UDP 등) 규격에 맞게 해석합니다.
- Preprocessor (전처리):** 파편화된 패킷을 재조립(Reassembly)하거나, 세션 관련 데이터를 처리하여 탐지 엔진이 분석하기 좋은 상태로 만듭니다.
- Detection (탐지):** 해독 및 전처리가 완료된 데이터를 사전에 정의된 공격 시그니처(Signature)와 비교하여 침입 여부를 결정하고 결과를 출력합니다.

실무 팁: 대규모 트래픽 환경에서는 데이터 수집 단계에서의 병목을 줄이기 위해 Raw Data 기반의 빠른 감지 방식을 고려해야 하며, DAQ 계층에서 메모리로 데이터를 얼마나 효율적으로 넘겨주는냐가 핵심입니다.

3. 성능 최적화 기술: Lua 및 Lua JIT

Snort와 같은 고성능 보안 장비에서는 규칙(Rule)의 유연성을 높이기 위해 스크립트 언어를 사용하기도 합니다. 이때 발생하는 성능 저하를 막기 위해 Lua JIT(Just-In-Time) 기술이 활용됩니다.

- Lua:** 가볍고 빠른 스크립트 언어로, 보안 로직이나 규칙을 동적으로 처리할 때 사용됩니다.
- Lua JIT의 원리:** 일반적인 인터프리터 방식과 달리, 프로그램 실행 시점에 필요한 부분만 미리 컴파일하여 기계어로 변환합니다. 이를 통해 스크립트 언어의 유연성을 유지하면서도 실행 속도를 획기적으로 높일 수 있습니다.

4. 운영 점검 리스트 (Operational Checklist)

시스템 구축 및 운영 단계에서 성능 저하나 보안 공백을 방지하기 위해 다음 항목을 반드시 점검해야 합니다.

[배치 및 구성 점검]

- [] **서비스 가용성 우선:** 네트워크 중단이 치명적인 환경인가? 그렇다면 IPS보다는 IDS(포트 미러링 방식) 배치를 우선 고려했는가?
- [] **차단 정책 확인:** IPS 모드 적용 시, 오탐(False Positive)으로 인해 정상적인 서비스 패킷이 Drop될 위험은 없는가?

[성능 및 최적화 점검]

- [] **DAQ 효율성:** 현재 환경에서 가장 적합한 DAQ 라이브러리(libpcap, afpacket 등)를 선택했는가?
- [] **처리 지연 시간:** IPS 운영 시, 패킷의 Decoder 및 Preprocessor 단계에서 발생하는 지연이 네트워크 서비스 수준 협약(SLA)을 충족하는가?
- [] **컴파일 최적화:** 스크립트 기반의 규칙을 사용할 경우, Lua JIT와 같은 최적화 기술이 적용되어 있는가?

[흔한 실수 및 주의사항]

- **오해:** "IDS는 네트워크 속도에 영향을 주지 않으니 항상 안전하다." → **사실:** IDS 자체는 트래픽 흐름을 막지 않지만, 분석을 위한 데이터 복사(Mirroring) 과정에서 스위치 부하가 발생할 수 있습니다.
- **실수:** IPS 구축 시 트래픽 처리량(Throughput) 계산을 간과하는 경우. → **결과:** 피크 타임(Peak Time)에 Snort의 처리 속도가 네트워크 속도를 따라가지 못하면 전체 네트워크가 마비되거나 패킷이 유실될 수 있습니다.