

수업 내용 정리 보고서

작성일: 2026년 6월 29일

챕터 구성

- 패킷 분석 도구의 활용과 트래픽 모니터링
- 패킷 조작 및 프로토콜 생성 기술
- 네트워크 공격 기법 및 스푸핑의 원리
- 실습 검증 및 보안 점검 체크리스트

패킷 분석 도구의 활용과 트래픽 모니터링

네트워크 보안과 트러블슈팅의 핵심은 네트워크를 통해 흐르는 데이터의 최소 단위인 **패킷(Packet)**을 정확히 식별하고 분석하는 것입니다. 패킷 분석은 단순히 데이터를 보는 것을 넘어, 프로토콜의 정상 동작 여부, 비정상적인 트래픽의 패턴, 그리고 공격자의 침입 흔적을 찾아내는 과정입니다. 본 장에서는 CLI 환경의 tcpdump부터 GUI 기반의 Wireshark, 그리고 파이썬 기반의 Scapy를 활용한 패킷 생성 및 공격 시뮬레이션까지의 과정을 다룹니다.

1. CLI 기반 패킷 캡처: tcpdump

tcpdump는 명령줄 인터페이스(CLI) 환경에서 동작하는 강력한 패킷 분석 프로그램입니다. 별도의 그래픽 화면 없이 터미널에서 즉각적으로 네트워크 트래픽을 캡처하고 필터링할 수 있어, 원격 서버 관리나 시스템 리소스가 제한된 환경에서 필수적으로 사용됩니다.

주요 옵션 및 사용법:

- D: 시스템에 존재하는 네트워크 인터페이스 카드(NIC) 목록을 확인합니다.
- i [인터페이스이름]: 특정 인터페이스(예: ens160)를 지정하여 패킷을 수집합니다.
- n: 호스트 이름을 IP 주소로 표시합니다.
- nn: 서비스 이름(예: ssh, http)을 포트 번호(예: 22, 80)로 직접 표시하여 변환 과정을 생략하고 정확한 포트 정보를 확인합니다.
- w [파일명]: 캡처한 패킷을 텍스트가 아닌 바이너리 파일(예: .pcap)로 저장합니다. 이는 나중에 Wireshark와 같은 GUI 도구로 정밀 분석할 때 사용됩니다.

패킷 필터링 기법:

방대한 트래픽 중 원하는 정보만 추출하기 위해 필터링 조건을 조합할 수 있습니다. src(출발지)와 dst(목적지)를 지정하거나, and 연산자를 사용하여 조건을 동시에 만족하는 패킷만 조회할 수 있습니다. 예를 들어, icmp and src host 172.16.16.1 명령은 출발지 IP가 172.16.16.1인 ICMP 패킷만을 선별합니다.

TCP 플래그(Flag) 해석:

TCP 통신 과정에서 패킷의 상태를 나타내는 플래그를 확인하는 것은 매우 중요합니다. S(SYN)는 연결 요청, .(ACK)는 수신 확인, P(PUSH)는 데이터 즉시 전달을 의미합니다. 만약 패킷에 이러한 TCP 관련 플래그가 나타나지 않는다면, 해당 통신은 비연결성 프로토콜인 UDP 통신일 가능성이 높습니다.

실제 HTTP 통신이 진행될 때 tcpdump를 통해 확인되는 로그는 다음과 같습니다. 각 패킷의 출발지/목적지 IP, 사용된 포트, 그리고 데이터 전송 상태를 나타내는 플래그를 확인할 수 있습니다.

tcpdump 출력 화면의 HTTP 프로토콜 명칭 및 Flags [P.] 값 확인

```
0:26:59.358870 IP 172.16.16.97.8080 > 172.16.20.1.59121: Flags [P.], seq 1584:1689, ck 1, win 495, length 105: HTTP
0:26:59.361912 IP 172.16.16.97.8080 > 172.16.20.1.62962: Flags [P.], seq 328:3248, ack 1849, win 448, length 2920: HTTP: HTTP/1.1 200 OK
0:26:59.361967 IP 172.16.16.97.8080 > 172.16.20.1.62962: Flags [P.], seq 3248:6168, ck 1849, win 448, length 2920: HTTP
0:26:59.362000 IP 172.16.16.97.8080 > 172.16.20.1.62962: Flags [P.], seq 6168:6275, ck 1849, win 448, length 107: HTTP
0:26:59.362656 IP 172.16.20.1.62962 > 172.16.16.97.8080: Flags [.], ack 3248, win 103, length 0
0:26:59.362855 IP 172.16.20.1.62962 > 172.16.16.97.8080: Flags [.], ack 6275, win 103, length 0
0:26:59.405951 IP 172.16.20.1.59121 > 172.16.16.97.8080: Flags [.], ack 1689, win 25, length 0
0:26:59.405960 IP 172.16.16.97.8080 > 172.16.20.1.59121: Flags [P.], seq 1689:2411, ck 1, win 495, length 722: HTTP
0:26:59.411563 IP 172.16.16.1.60519 > 172.16.16.97.8080: Flags [.], ack 1689, win 25, length 0
0:26:59.411573 IP 172.16.16.97.8080 > 172.16.16.1.60519: Flags [P.], seq 1689:2411, ck 1, win 495, length 722: HTTP
0:26:59.411621 IP 172.16.16.1.60519 > 172.16.16.97.8080: Flags [.], ack 1689, win 25, length 0
0:26:59.451838 IP 172.16.20.1.59121 > 172.16.16.97.8080: Flags [.], ack 2411, win 25, length 0
0:26:59.456452 IP 172.16.16.1.60519 > 172.16.16.97.8080: Flags [.], ack 2411, win 25, length 0
0:26:59.456459 IP 172.16.16.1.60519 > 172.16.16.97.8080: Flags [.], ack 2411, win 25, length 0
0:26:59.607527 IP 172.16.20.1.62962 > 172.16.16.97.8080: Flags [P.], seq 1849:2768, ck 6275, win 1023, length 919: HTTP: GET /api/kernelspecs?1782696319368 HTTP/1.1
0:26:59.607843 IP 172.16.20.1.53591 > 172.16.16.97.8080: Flags [P.], seq 917:1027, ack 328, win 1022, length 910: HTTP: GET /api/me?1782696319368 HTTP/1.1
0:26:59.608843 IP 172.16.16.97.8080 > 172.16.20.1.62962: Flags [P.], seq 6275:7140, ck 2768, win 448, length 865: HTTP: HTTP/1.1 200 OK
0:26:59.609494 IP 172.16.16.97.8080 > 172.16.20.1.53591: Flags [P.], seq 328:934, ack 1827, win 452, length 606: HTTP: HTTP/1.1 200 OK
0:26:59.652432 IP 172.16.20.1.62962 > 172.16.16.97.8080: Flags [.], ack 7140, win 100, length 0
0:26:59.652447 IP 172.16.20.1.53591 > 172.16.16.97.8080: Flags [.], ack 934, win 102, length 0
```

tcpdump 명령어를 통해 수집된 HTTP 프로토콜 패킷의 통신 로그 및 TCP 플래그 상태

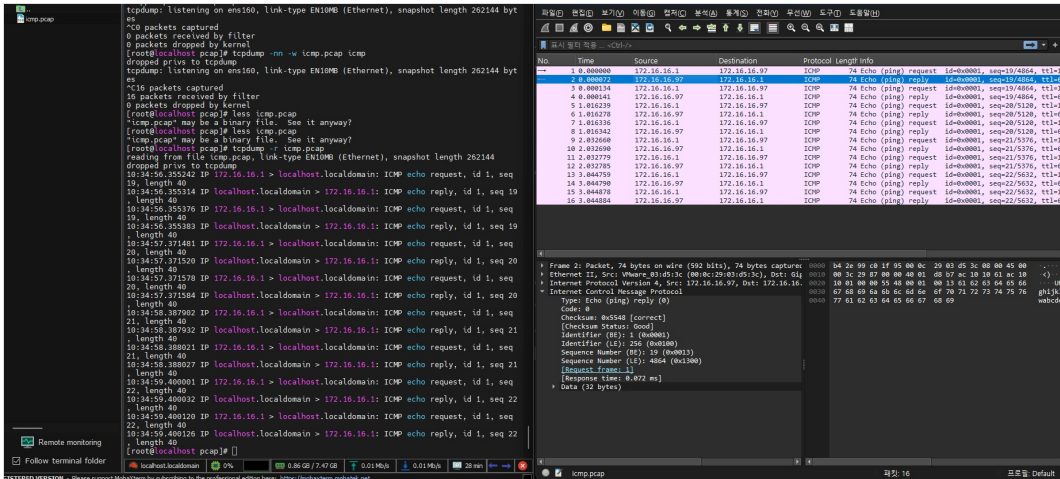
위 로그를 통해 172.16.16.97 호스트가 172.16.20.1 호스트로 HTTP 데이터를 전송하고 있으며, [P.] 플래그를 통해 데이터가 PUSH 및 ACK 상태임을 확인할 수 있습니다.

2. 프로토콜 분석기: Wireshark

Wireshark는 tcpdump와 달리 그래픽 사용자 인터페이스(GUI)를 제공하는 전문 프로토콜 분석기입니다. tcpdump로 캡처한 .pcap 파일을 불러와 패킷의 계층 구조(Ethernet, IP, TCP/UDP, Application Layer)를 시각적으로 상세하게 분석할 수 있습니다.

실습에서는 tcpdump를 사용하여 ICMP 패킷을 icmp.pcap 파일로 저장한 후, 이를 Wireshark로 열어 검증하는 절차를 수행합니다. 터미널에서 tcpdump -nn -w icmp.pcap icmp 명령을 실행하여 ICMP 트래픽을 파일로 저장하고, Wireshark의 패킷 리스트와 상세 정보 창을 통해 각 패킷의 세부 필드(Identifier, Sequence Number 등)를 확인합니다.

터미널의 tcpdump 파일 저장 명령과 wireshark의 ICMP 패킷 목록 및 상세 정보



이 절의 실습 결과를 확인하는 화면

이 화면은 터미널의 캡처 로그와 Wireshark의 분석 화면을 동시에 보여줍니다. Wireshark를 통해 ICMP Echo Request와 Reply가 정상적으로 오가고 있음을 시각적으로 검증할 수 있습니다.

3. 네트워크 매핑 정보 확인: ARP

IP 주소는 논리적인 주소이며, 실제 데이터 전송을 위해서는 물리적인 주소인 **MAC 주소**가 필요합니다. **ARP(Address Resolution Protocol)**는 IP 주소를 통해 해당 호스트의 MAC 주소를 찾아내는 역할을 합니다. 보안 실습에서는 ARP 스푸핑(Spoofing)을 통해 IP와 MAC 주소의 매핑 정보를 변조하여 중간자 공격(MITM)이나 DoS 공격을 수행할 수 있습니다.

시스템의 ARP 테이블을 확인하면 현재 네트워크 상에서 통신 중인 각 IP 주소에 대응하는 물리적 MAC 주소 목록을 볼 수 있습니다.

Internet Address와 Physical Address 항목에 매핑된 IP 및 MAC 주소 값

Interface	172.16.20.1	---	0x8
Internet Address	IP	Physical Address	Type
172.16.0.1	eth0	b0-38-6c-3d-0e-2a	dynamic
172.16.1.1	Ethernet	70-85-c2-64-36-4d	dynamic
172.16.4.2	eth0	18-c0-4d-24-6e-e8	dynamic
172.16.16.97	eth0	00-0c-29-03-d5-3c	dynamic
172.16.20.100	eth0	00-0c-29-f7-09-1a	dynamic
172.16.122.3	eth0	56-10-3d-a5-c8-12	dynamic
172.16.123.162	Ethernet	18-c0-4d-24-6f-19	dynamic

IP 주소와 이에 대응하는 물리적 MAC 주소 매핑 정보를 나타내는 ARP 테이블

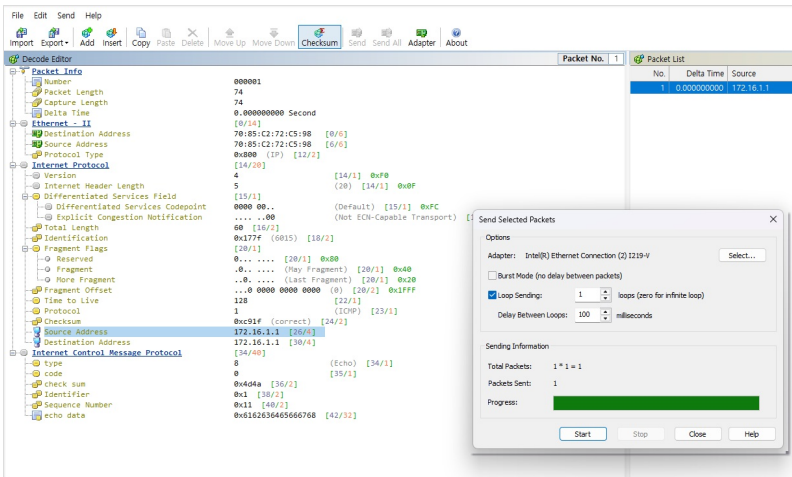
ARP 테이블 결과에서 172.16.1.1 IP 주소는 70-85-c2-64-36-4d라는 물리적 MAC 주소와 매핑되어 있음을 확인할 수 있습니다.

4. 패킷 조작 및 공격 시뮬레이션

정상적인 패킷 외에 공격자가 의도적으로 조작한 패킷을 생성하는 도구들을 학습합니다.

Packetbuilder: 패킷에 담긴 페이로드(Payload)나 헤더 정보를 임의로 수정할 수 있는 프로그램입니다. 이를 통해 소스(Source) IP를 변조하여 자신의 정체를 숨기는 **IP 스푸핑(Spoofing)**을 수행할 수 있으며, 이는 분산 서비스 거부 공격(DDoS)의 기초가 됩니다.

Decode Editor의 프로토콜 상세 트리과 Send Selected Packets 설정 화면



packetbuilder를 이용한 IP/ICMP 계층 구조 설정 및 패킷 전송 옵션 구성 화면

Packetbuilder를 사용하여 Ethernet, IP, ICMP 계층의 세부 정보를 설정하고, 루프(Loop) 횟수 및 지연 시간을 조절하여 특정 패킷을 반복 전송할 수 있습니다.

주요 공격 유형:

- **DoS (Denial of Service):** 서비스 거부 공격. 대상 시스템의 자원을 고갈시켜 정상적인 서비스를 불가능하게 만듭니다.
- **DDoS (Distributed DoS):** 여러 대의 좀비 PC를 이용해 동시에 공격하는 분산 서비스 거부 공격입니다.
- **RDDOS (Reflected DDoS):** 반사 공격. 공격 대상(dst)을 DNS 서버(예: 8.8.8.8)와 같은 공개 서버로 설정하여, 서버가 공격 대상에게 대량의 응답을 보내도록 유도하는 방식입니다.
- **Land Attack:** 출발지(src)와 목적지(dst) IP 및 포트를 모두 피해자 자신의 주소로 변조하여, 대상 시스템이 자기 자신에게 무한 루프를 돌게 만드는 공격입니다.

Scapy (Python Module): Python 프로그래밍을 통해 패킷을 직접 설계(Build)하고 전송(Play)할 수 있는 강력한 라이브러리입니다. 프로토콜의 캡슐화(Encapsulation) 과정을 코드로 직접 제어할 수 있어 정교한 공격 스크립트 작성이 가능합니다.

Scapy를 이용한 ICMP 패킷 생성 코드는 다음과 같이 작성됩니다. `IP(dst=target_ip)` / `ICMP()`와 같이 슬래시(/) 연산자를 사용하여 계층 간 캡슐화를 구현합니다.

scapy 모듈 임포트 및 IP와 ICMP 프로토콜을 결합하여 packet 객체를 생성하는 코드

```
from scapy.all import IP, ICMP, send

target_ip = "172.16.1.1"

packet = IP(dst=target_ip) / ICMP()

#send(packet, count = 100, verbose = 1)
print(packet.summary())
print(packet.show())
```

scapy 모듈을 사용하여 특정 대상 IP로 전송할 ICMP 패킷을 정의하는 파이썬 코드

코드에서 `target_ip`를 172.16.1.1로 설정하고, ICMP 프로토콜을 결합하여 패킷 객체를 생성하는 과정을 볼 수 있습니다.

실제로 공격을 수행하기 위해서는 주석 처리된 `send()` 함수를 활성화해야 합니다. `count=100` 옵션을 주면 해당 패킷을 100번 반복하여 전송함으로써 대상 시스템에 부하를 주는 DoS 시도를 수행할 수 있습니다.

`send` 함수 앞의 주석(#)이 제거되어 실행 가능한 상태인 코드

```
from scapy.all import IP, ICMP, send

target_ip = "172.16.1.1"

packet = IP(dst=target_ip) / ICMP()

send(packet, count = 100, verbose = 1)
```

작성된 scapy 스크립트의 `send` 함수 주석을 해제하여 ICMP 패킷 전송을 실행하는 단계

`send(packet, count=100, verbose=1)` 명령을 통해 172.16.1.1을 대상으로 100개의 ICMP 패킷을 전송하는 실습이 진행됩니다.

실무 및 학습 점검 포인트

- **도구 선택 기준:** 단순 트래픽 모니터링 및 필터링은 `tcpdump`가 효율적이며, 생성된 패킷의 계층별 상세 구조를 정밀 분석할 때는 `Wireshark`를 사용하는 것이 적절합니다.
- **TCP 상태 확인:** 통신 장애 발생 시, TCP 플래그(SYN, ACK, FIN 등)를 확인하여 연결 과정의 어느 단계에서 문제가 발생했는지(예: 3-Way Handshake 실패 여부) 판단해야 합니다.
- **ARP 보안 주의사항:** 네트워크 내에서 IP와 MAC 주소가 불일치하거나, 동일한 MAC 주소가 여러 IP에 할당되어 있다면 ARP 스푸핑 공격을 의심해야 합니다.
- **Scapy 활용 시 주의:** Scapy를 통한 패킷 생성 및 전송 실습은 반드시 격리된 테스트 환경(가상 네트워크 등)에서 수행해야 합니다. 승인되지 않은 네트워크에 대한 DoS 시도는 법적 문제가 될 수 있습니다.

패킷 조작 및 프로토콜 생성 기술

네트워크 보안을 이해하기 위해서는 네트워크 상에서 흐르는 데이터를 직접 관찰하고, 이를 인위적으로 조작하여 다양한 공격 시나리오를 재현해보는 과정이 필수적입니다. 본 장에서는 CLI 환경의 `tcpdump`부터 GUI 기반의 `Wireshark`, 그리고 파이썬 라이브러리인 `Scapy`를 활용하여 패킷을 분석하고 생성 및 변조하는 기술을 다룹니다.

1. CLI 기반 패킷 분석: tcpdump

`tcpdump`는 명령행 인터페이스(CLI)에서 동작하는 강력한 패킷 분석 도구입니다. 네트워크 인터페이스(NIC)를 통해 통과하는 패킷을 캡처하여 실시간으로 보여 주거나 파일로 저장할 수 있습니다.

- **주요 옵션:**
 - `-D`: 시스템에 존재하는 네트워크 인터페이스(NIC) 목록을 확인합니다.

- `-i` [인터페이스명]: 분석할 특정 인터페이스(예: `ens160`)를 지정합니다.
 - `-n`: 호스트 이름을 IP 주소로 표시합니다.
 - `-nn`: 서비스 이름(예: `ssh`) 대신 포트 번호(예: `22`)를 직접 표시하여 분석의 정확성을 높입니다.
 - `-w` [파일명]: 캡처한 패킷을 파일로 저장합니다.
- **필터링 기법**: 특정 프로토콜(예: `icmp`), 출발지(`src`), 목적지(`dst`)를 지정할 수 있으며, `and` 연산자를 사용하여 여러 조건을 동시에 적용할 수 있습니다. (예: `icmp and src host 172.16.16.1`)
 - **TCP Flag 확인**: TCP 통신 시 패킷의 상태를 나타내는 플래그를 통해 연결 상태를 파악합니다. `S(SYN)`, `.(ACK)`, `P(PUSH)` 등이 대표적입니다. 만약 플래그 정보가 없다면 UDP 통신일 가능성이 높습니다.

실제 통신 환경에서 `tcpdump`를 실행하면 다음과 같이 HTTP 프로토콜의 흐름과 TCP 플래그 상태를 확인할 수 있습니다. 로그를 보면 `172.16.16.97` 서버가 `172.16.20.1` 클라이언트에게 데이터를 전송(`PUSH` 플래그)하고, 클라이언트가 이에 대해 응답(`ACK` 플래그)하는 과정을 볼 수 있습니다.

로그 분석 시 `Flags [P.]`와 같은 표기는 데이터가 전송(`PUSH`)되었고, 해당 패킷이 수신되었음을 확인(`ACK`)했다는 의미이므로 통신이 정상적으로 이루어지고 있음을 나타냅니다.

2. 패킷 캡처 및 프로토콜 상세 분석

네트워크 트래픽을 분석할 때는 실시간 관찰뿐만 아니라, 캡처된 파일을 나중에 정밀하게 검토하는 과정이 중요합니다. `tcpdump -w` 옵션으로 저장한 `.pcap` 파일은 **Wireshark**와 같은 전문 프로토콜 분석기를 통해 계층별 구조를 시각적으로 파악할 수 있습니다.

실습에서는 `icmp` 프로토콜을 대상으로 패킷을 캡처하여 파일로 저장한 후, 이를 다시 `tcpdump -r` 명령어로 읽거나 **Wireshark**를 통해 분석하는 과정을 수행합니다. **Wireshark**를 사용하면 `ICMP Echo Request(요청)`와 `Echo Reply(응답)`가 어떤 IP 사이에서 오가는지 한눈에 확인할 수 있습니다.

Wireshark의 패킷 리스트 창에서는 시간, 소스, 목적지, 프로토콜, 길이 등의 정보를 제공하며, 상세 창을 통해 Ethernet 계층부터 IP, ICMP 계층까지의 필드 값을 정밀하게 검증할 수 있습니다.

3. L2/L3 주소 매핑 및 스누핑의 기초

패킷이 목적지까지 전달되기 위해서는 IP 주소(L3)뿐만 아니라 물리적 주소인 MAC 주소(L2)를 알아야 합니다. 컴퓨터는 **ARP(Address Resolution Protocol)**를 통해 IP 주소에 대응하는 MAC 주소를 확인하고 이를 ARP 테이블에 저장합니다.

ARP 테이블을 확인하면 특정 IP 주소가 어떤 물리적 주소(Physical Address)와 매핑되어 있는지 알 수 있습니다. 이는 네트워크 공격인 **ARP 스누핑**을 이해하는 기초가 됩니다. 공격자가 자신의 MAC 주소를 피해자의 IP와 매핑되도록 속이면, 네트워크 트래픽을 중간에서 가로채는 `DoS` 공격이 가능해집니다.

테이블에서 `172.16.1.1` 주소에 대해 `70-85-c2-64-36-4d`라는 물리적 주소가 할당되어 있음을 확인할 수 있습니다.

4. 패킷 조작을 통한 공격 시나리오

패킷의 페이로드(Payload)나 헤더 정보를 수정하면 정상적인 통신이 아닌 공격용 패킷을 생성할 수 있습니다. **PacketBuilder**와 같은 도구는 프로토콜의 각 필드를 직접 수정할 수 있는 환경을 제공합니다.

- **DoS (Denial of Service)**: 서비스를 거부하는 공격으로, 특정 대상의 자원을 고갈시킵니다.
- **DDoS (Distributed DoS)**: 여러 대의 좀비 PC를 이용하여 분산하여 공격하는 방식입니다.
- **RDDOS (Reflected DDoS)**: 공격 대상이 아닌 제3의 서버(예: DNS 서버 `8.8.8.8`)를 반사판으로 이용하여 공격을 수행하는 방식입니다.
- **Land Attack**: 소스(`src`) IP와 목적지(`dst`) IP를 모두 피해자의 IP로 변조하여, 자기 자신에게 패킷이 계속 돌아오게 만들어 시스템을 마비시키는 공격입니다.

PacketBuilder를 사용하면 패킷의 계층별 구조(Ethernet, IP, ICMP 등)를 상세히 설정할 수 있습니다. 특히 'Send Selected Packets' 기능을 통해 특정 횟수만큼 패킷을 반복 전송(Loop Sending)하도록 설정하여 `DoS` 공격을 시뮬레이션할 수 있습니다.

화면의 설정값을 보면 ICMP 프로토콜을 선택하고, Loop Sending을 1회로 설정하여 단일 패킷 전송을 준비하거나, 횟수를 늘려 대량의 패킷을 발생시키는 방식으로 조작이 가능합니다.

5. Scapy를 이용한 프로그래밍 기반 패킷 생성

도구를 사용하는 단계를 넘어, **Python**의 **Scapy** 모듈을 사용하면 프로그래밍 방식으로 매우 정교한 패킷을 생성(Build)하고 전송(Play)할 수 있습니다. **Scapy**의 핵심은 **Encapsulation(캡슐화)** 개념을 / 연산자로 구현한다는 점입니다. 예를 들어 `IP()/ICMP()`는 IP 헤더 안에 ICMP 헤더를 포함시키는 구조를 의미합니다.

먼저, **Scapy**를 사용하여 공격 대상의 IP를 설정하고 ICMP 패킷의 구조를 정의하는 코드를 작성합니다. 아래 코드는 `172.16.1.1`을 대상으로 하는 패킷을 생성하지만, 아직 전송 명령이 주석 처리되어 있어 실제 공격은 발생하지 않는 준비 단계입니다.

패킷 생성이 완료되면 `packet.show()` 명령을 통해 생성된 패킷의 계층 구조와 필드 값을 검증할 수 있습니다. 이후, 주석을 해제하고 `send()` 함수를 실행하면 설정된 횟수(예: `count=100`)만큼 패킷이 대상에게 즉시 전송됩니다.

실행 결과, 작성된 스크립트에 따라 지정된 타겟 IP로 대량의 ICMP 패킷이 전송되며, 이는 **Scapy**를 활용한 자동화된 `DoS` 공격의 기본 원리를 보여줍니다.

학습 요약 및 점검 사항

패킷 조작 기술은 네트워크의 동작 원리를 파악하는 동시에, 보안 위협을 방어하기 위한 필수 역량입니다. 다음 체크리스트를 통해 학습 내용을 점검하십시오.

- **도구 활용**: `tcpdump`의 옵션(`-nn`, `-w` 등)과 필터링 문법을 정확히 숙지하고 있는가?
- **플래그 분석**: TCP 세션 흐름에서 `SYN`, `ACK`, `PUSH` 플래그가 어떤 의미를 갖는지 구분할 수 있는가?

- **주소 매핑:** ARP 테이블을 통해 L2 주소와 L3 주소의 관계를 파악하고, 이를 이용한 스푸핑 원리를 이해했는가?
- **공격 메커니즘:** DoS, DDoS, Land Attack의 차이점과 각각의 패킷 변조 방식(src/dst 조작 등)을 설명할 수 있는가?
- **Scapy 프로그래밍:** IP() / ICMP()와 같은 캡슐화 문법을 사용하여 원하는 형태의 패킷을 코드로 설계할 수 있는가?

네트워크 공격 기법 및 스푸핑의 원리

네트워크 보안을 이해하기 위한 첫 번째 단계는 네트워크를 흐르는 데이터의 최소 단위인 패킷(Packet)을 정확히 분석하는 것입니다. 패킷 내부에 포함된 프로토콜 정보, IP 주소, MAC 주소, 그리고 제어 플래그(Flag)를 파악함으로써 정상적인 통신과 공격 의도가 담긴 비정상적인 통신을 구분할 수 있습니다.

1. 패킷 분석 도구 및 프로토콜 확인

패킷 분석은 목적에 따라 명령줄 인터페이스(CLI) 기반의 도구와 그래픽 사용자 인터페이스(GUI) 기반의 도구로 나뉩니다.

- **tcpdump:** CLI 환경에서 패킷을 캡처하고 분석하는 강력한 도구입니다.
 - **-D:** 시스템의 네트워크 인터페이스 카드(NIC) 목록을 조회합니다.
 - **-i** [인터페이스명]: 특정 인터페이스(예: ens160)를 지정하여 패킷을 확인합니다.
 - **-n:** 호스트네임을 IP 주소로 표시합니다.
 - **-nn:** 서비스 이름(예: ssh) 대신 포트 번호(예: 22)로 표시하여 분석의 명확성을 높입니다.
 - **-w** [파일명]: 캡처한 패킷을 파일로 저장하여 나중에 Wireshark 등으로 정밀 분석할 수 있게 합니다.
- **Wireshark:** 프로토콜 분석기(Protocol Analyzer)로, 캡처된 패킷을 계층별로 시각화하여 상세히 보여주는 GUI 도구입니다.

tcpdump를 통해 실시간으로 흐르는 HTTP 패킷을 확인하면, 출발지(src)와 목적지(dst) IP, 그리고 통신 상태를 나타내는 플래그(flag)를 볼 수 있습니다. 예를 들어, [P] 플래그는 PUSH를 의미하며, [.]는 ACK를 나타냅니다. 만약 이러한 TCP 플래그가 보이지 않는다면 해당 패킷은 UDP 통신일 가능성이 높습니다.

tcpdump 실행 결과에서 172.16.16.97 호스트가 172.16.20.1로 HTTP 데이터를 전송하며, PUSH(P) 플래그와 함께 시퀀스 번호(seq) 및 데이터 길이를 전달하는 것을 확인할 수 있습니다.

캡처된 패킷은 .pcap 확장자를 가진 파일로 저장할 수 있습니다. 저장된 파일은 tcpdump -r [파일명] 명령으로 다시 읽거나, Wireshark와 같은 도구를 사용하여 프로토콜 계층 구조를 상세히 분석하는 데 사용됩니다.

실습에서는 icmp.pcap 파일로 ICMP 패킷을 저장한 후, 터미널에서 less 명령어로 내용을 확인하고 Wireshark를 통해 ICMP Echo Request와 Reply의 흐름을 계층별로 검증하였습니다.

2. 계층 간 주소 매핑과 스푸핑의 기초

네트워크 공격, 특히 스푸핑(Spoofing)을 이해하려면 IP 주소(3계층)와 MAC 주소(2계층)의 관계를 알아야 합니다. ARP(Address Resolution Protocol)는 IP 주소를 기반으로 상대방의 물리적 주소인 MAC 주소를 확인하는 역할을 합니다.

공격자는 이 ARP 과정을 악용하여 자신의 MAC 주소를 피해자의 IP에 매핑시키거나, 그 반대의 방식으로 IP/MAC 스푸핑을 수행하여 데이터를 가로채거나 서비스를 방해할 수 있습니다. 아래 ARP 테이블은 현재 네트워크에서 확인되는 IP 주소와 그에 대응하는 물리적 MAC 주소의 매핑 정보를 보여줍니다.

3. 패킷 변조 및 생성 기술

정상적인 통신 외에 공격을 수행하기 위해서는 패킷의 페이로드(Payload)나 헤더 정보를 임의로 수정할 수 있는 도구가 필요합니다.

Packetbuilder는 패킷의 프로토콜 계층을 직접 설정하고 페이로드를 수정할 수 있는 프로그램입니다. 이를 통해 소스 IP를 변조하여 자신의 신원을 숨기거나, 특정 대상에게 대량의 패킷을 보내는 공격을 준비할 수 있습니다. 사용자는 인터페이스 선택, 루프 전송(Loop Sending) 횟수, 전송 간격(Delay) 등을 설정하여 패킷을 생성 및 전송할 수 있습니다.

보다 정교하고 자동화된 패킷 생성 및 플레이를 위해서는 Python의 **Scapy** 모듈을 활용합니다. Scapy는 캡슐화(Encapsulation) 원리를 이용하여 프로토콜을 자유롭게 조합할 수 있는 강력한 라이브러리입니다.

Scapy를 사용하면 다음과 같은 절차로 패킷을 정의합니다. 먼저 'IP' 계층에 목적지 주소(dst)를 지정하고, 그 위에 'ICMP' 프로토콜을 결합(/ 연산자 사용)하여 패킷 객체를 생성합니다.

패킷 정의가 완료되면, 'send()' 함수를 호출하여 실제 네트워크로 패킷을 방출합니다. 이때 'count' 인자를 사용하여 전송 횟수를 지정함으로써 특정 대상에게 지속적인 트래픽을 보내는 DoS 공격을 시뮬레이션할 수 있습니다.

4. 주요 네트워크 공격 기법

패킷 변조와 스푸핑 기술이 결합되면 다음과 같은 다양한 서비스 거부 공격(DoS)이 가능해집니다.

- **DoS (Denial of Service):** 단일 공격자가 특정 시스템의 자원을 고갈시켜 정상적인 서비스를 불가능하게 만드는 공격입니다.
- **DDoS (Distributed Denial of Service):** 여러 대의 공격 장비(좀비 PC 등)를 동원하여 분산된 상태로 공격을 수행하므로 차단이 훨씬 어렵습니다.
- **RDDOS (Reflected DDoS):** 공격자가 소스 IP를 피해자의 IP로 변조하여 DNS 서버(예: 8.8.8.8)와 같은 공용 서비스에 요청을 보냅니다. 서비스 서버는 변조된 소스 IP(피해자)로 응답을 보내게 되며, 이 과정에서 대량의 응답 트래픽이 피해자에게 집중되는 현상을 이용합니다.
- **Land Attack:** 소스 IP(src)와 목적지 IP(dst)를 모두 피해자의 IP로 변조하여 패킷을 보냅니다. 피해 시스템은 자기 자신에게 응답을 보내려는 무한 루프에 빠지게 되어 시스템이 마비될 수 있습니다.

결론적으로, IP/MAC 스푸핑은 이러한 DoS 및 DDoS 공격의 핵심적인 수단으로 활용되며, 이를 방어하기 위해서는 ARP 보안(Dynamic ARP Inspection 등)과 패킷의 비정상적인 헤더 패턴을 탐지하는 능력이 필수적입니다.

[학습 점검 및 실무 포인트]

- **패킷 분석 검증:** tcpdump 사용 시 -nn 옵션을 누락하면 호스트네임 해석 과정에서 지연이 발생하거나 분석이 혼란스러울 수 있으므로 주의해야 합니다.
- **공격 시나리오 이해:** Land Attack은 '자기 자신에게 보내는 패킷'이라는 특성을 이해하는 것이 핵심입니다.
- **Scapy 활용:** 코드 작성 시 packet = IP(dst="target") / ICMP()와 같이 슬래시(/)를 사용하여 계층을 결합하는 문법을 숙지하십시오.
- **보안 주의사항:** Scapy를 이용한 패킷 전송 실험은 반드시 격리된 가상 환경에서 수행해야 하며, 실제 네트워크 환경에서의 시도는 법적 문제가 될 수 있습니다.

실습 검증 및 보안 점검 체크리스트

본 절에서는 tcpdump와 Wireshark를 활용한 패킷 분석, ARP 테이블을 통한 매핑 확인, 그리고 Packetbuilder와 Scapy를 이용한 패킷 변조 및 DoS(Denial of Service) 공격 시뮬레이션 과정을 검증한다. 실습 과정에서 확인된 데이터는 네트워크 보안 상태를 점검하고 공격 메커니즘을 이해하는 핵심 근거로 활용된다.

1. 패킷 캡처 및 프로토콜 분석 검증

네트워크 트래픽을 실시간으로 모니터링하기 위해 CLI 기반의 tcpdump를 사용한다. tcpdump는 특정 인터페이스(예: ens160)를 지정하여 패킷을 수집하며, 옵션을 통해 분석 효율을 높일 수 있다. 특히 -nn 옵션은 호스트네임이나 서비스 이름 대신 IP 주소와 포트 번호를 그대로 표시하여 분석의 명확성을 제공한다. 패킷의 플래그(flag)를 확인하면 현재 통신이 연결(SYN), 확인(ACK), 데이터 전송(PUSH) 중 어느 단계에 있는지 파악할 수 있다.

실습에서는 tcpdump 명령어를 통해 HTTP 프로토콜의 흐름을 추적하였다. 로그를 살펴보면 172.16.16.97에서 172.16.20.1로 전송되는 데이터의 플래그 상태가 [P.] (PUSH/ACK)로 나타나며, 이는 세션이 유지된 상태에서 데이터가 전송되고 있음을 의미한다.

로그 분석 결과, 데이터의 길이(length)와 시퀀스 번호(seq)가 정상적으로 증가하며 통신이 이루어지고 있음을 확인할 수 있다. 만약 플래그가 나타나지 않는다면 이는 비연결성 프로토콜인 UDP 통신일 가능성이 높으므로 주의가 필요하다.

패킷 분석의 정밀도를 높이기 위해 tcpdump로 캡처한 파일(.pcap)을 프로토콜 분석기인 Wireshark로 로드하여 상세 계층 구조를 확인한다. tcpdump에서 -w 옵션을 사용하여 파일로 저장한 뒤 Wireshark에서 열면, 텍스트 기반의 로그를 시각적인 계층 구조로 상세히 분석할 수 있다.

실습에서는 ICMP 패킷을 대상으로 tcpdump를 통해 icmp.pcap 파일을 생성하였고, 이를 Wireshark로 불러와 Echo (ping) request와 reply가 정상적으로 교환되는 것을 검증하였다.

분석 결과, Source와 Destination IP 간의 ICMP 요청과 응답이 시간 순서대로 일치하게 나타나며, 이는 네트워크 계층에서의 기본적인 도달 가능성 (Reachability)이 확보되었음을 의미한다.

2. 주소 매핑 및 스푸핑 점검

네트워크 보안의 핵심 중 하나는 IP 주소와 물리적 주소인 MAC 주소 사이의 일관성을 유지하는 것이다. ARP(Address Resolution Protocol)는 IP 주소를 기반으로 상대방의 MAC 주소를 찾아내는 역할을 한다. 만약 공격자가 ARP Spoofing을 통해 자신의 MAC 주소를 피해자의 IP에 매핑시킨다면, 트래픽을 가로채거나 DoS 공격을 수행할 수 있다.

실습을 통해 인터페이스에 등록된 ARP 테이블을 확인하여, 각 IP 주소에 할당된 물리적 주소가 의도한 대로 매핑되어 있는지 점검하였다.

ARP 테이블 확인 결과, 172.16.1.1 주소에 70-85-c2-64-36-4d와 같은 특정 MAC 주소가 정상적으로 매핑되어 있다. 보안 점검 시에는 동일한 IP에 대해 MAC 주소가 빈번하게 변경되거나, 게이트웨이의 MAC 주소가 알 수 없는 주소로 바뀌는지 반드시 확인해야 한다.

3. 패킷 변조 및 공격 시뮬레이션 검증

공격자는 패킷의 페이로드나 헤더 정보를 수정하여 공격을 수행한다. Packetbuilder와 같은 도구는 패킷의 소스 IP를 변조(Spoofing)하여 공격자의 신원을 숨기거나, LAND Attack(출발지/목적지 IP를 모두 피해자로 설정)을 수행하는 데 사용될 수 있다.

Packetbuilder의 Decode Editor를 사용하면 패킷의 각 계층(Ethernet, IP, ICMP 등)을 상세히 수정할 수 있다. 실습에서는 ICMP 패킷의 세부 설정을 확인하고, Send Selected Packets 옵션을 통해 변조된 패킷을 특정 횟수만큼 반복 전송하는 과정을 수행하였다.

화면을 통해 확인할 수 있듯이, 패킷의 Source Address를 임의로 설정할 수 있으며, Loop Sending 기능을 활용하여 지속적인 패킷 유입을 통한 DoS 상황을 모사할 수 있다.

보다 자동화되고 정교한 공격 시뮬레이션을 위해 파이썬의 Scapy 모듈을 활용한다. Scapy는 패킷의 생성(Build)과 전송(Play)을 프로그래밍 방식으로 수행할 수 있게 하며, encapsulation을 통해 복잡한 프로토콜 계층을 쉽게 구성할 수 있다.

먼저, 파이썬 스크립트를 통해 대상 IP(172.16.1.1)로 ICMP 패킷을 생성하는 코드를 작성한다. IP(dst=target_ip) / ICMP() 구문은 IP 계층과 ICMP 계층을 결합(encapsulation)하는 과정을 의미한다.

작성된 코드에서 주석 처리된 send 함수를 해제하여 실제 패킷 전송을 실행한다. count=100 옵션은 100개의 패킷을 연속적으로 전송하도록 설정하여 DoS 공격의 형태를 구현한다.

실행 결과, 지정된 횟수만큼 패킷이 전송됨으로써 Scapy를 이용한 패킷 생성 및 전송 기능이 정상적으로 동작함을 확인하였다. 이러한 기법은 DDoS 공격 중 하나인 RDDoS(Reflected DDoS), 즉 DNS 서버 등을 반사판으로 활용하여 공격하는 방식의 기초 원리가 된다.

보안 점검 체크리스트

실습 및 실제 환경 운영 시 다음 항목을 주기적으로 점검하여 네트워크 보안성을 유지해야 한다.

점검 영역	점검 항목	판단 기준 및 확인 방법
패킷 무결성	프로토콜 플래그 정상 여부	tcpdump 분석 시 비정상적인 플래그 조합이나 의도치 않은 PUSH 플래그 과다 발생 여부 확인
주소 변조 방지	ARP 테이블 일관성	arp -a 명령어를 통해 게이트웨이 및 주요 서버의 IP-MAC 매핑이 변경되지 않았는지 주기적 대조
공격 탐지	DoS/DDoS 트래픽 패턴	특정 IP로부터 비정상적으로 높은 빈도의 ICMP/UDP 패킷이 유입되는지 모니터링
변조 공격 대응	IP Spoofing 여부	패킷 헤더의 Source IP가 내부 네트워크 대역과 일치하는지, 또는 비정상적인 외부 IP인지 검증