BEGINNER-TO-PRACTITIONER · CONCEPT & MECHANISM EDITION

정보보안·네트워크 개념과 메커니즘 완전 해설서

비전공자가 단어의 뜻부터 장비·운영체제 내부 동작, 설정 명령, 장애와 보안 문제까지 한 흐름으로 이해하도록 다시 쓴 통합 교재

용어를 문장으로 정의
"Port Security란 무엇인가", "802.1Q란 무엇인가"를 독립 절에서 먼저 설명

원인부터 결과까지 추적
L2 loop의 frame 복제 → MAC flapping → broadcast storm → STP 차단 과정을 단계별 해설

그림을 해설의 일부로 사용
header, table lookup, state transition, packet path, 공격·방어 지점을 번호로 연결

필기 밖의 기반 지식 보충
표준·공식 문서를 바탕으로 빠진 전제와 실무 연결을 분리해 추가

제4판 · 단어·개념·메커니즘 중심 전면 재작성 · 2026년 7월
기반 자료: 제공된 75일 과정 필기와 키워드 정리 · 대상: 네트워크·정보보안 비전공 입문자

이 책을 다시 쓴 이유와 읽는 방법

이 책은 “키워드가 어디에 들어가는가”를 정리한 색인이 아니다. 단어 하나를 보고도 그 말이 가리키는 대상, 등장한 이유, 장비와 운영체제 내부의 처리 순서, 정상 상태와 장애 상태의 차이, 확인 명령과 출력 해석까지 설명할 수 있도록 만든 개념 해설서다. 원본 75일 필기와 키워드 정리는 실제 수업에서 접한 범위를 확인하는 기준으로 사용했고, 필기에 빠진 전제는 공식 표준과 프로젝트 문서를 통해 보충했다.

본문의 서술 단위

핵심 용어는 가능한 한 **“X란 …이다”**라는 정의 문장으로 시작한다. 이어서 ① 왜 필요한가, ② 내부에서 무엇을 읽고 바꾸는가, ③ 순서대로 어떻게 동작하는가, ④ 실패하면 어떤 현상이 보이는가, ⑤ 어떤 명령과 packet capture로 확인하는가, ⑥ 어떤 보안 통제가 연결되는가를 설명한다.

표시	의미	읽는 방법
필기 확인 + 보충 해설	원본 필기에 등장한 개념을 중심으로 표준 동작을 확장	수업 기억을 되살린 뒤 세부 동작을 연결한다.
기초 보강 / OS·RFC·IEEE 보강	필기에 명시되지 않았지만 다음 개념을 이해하는 데 필요한 전제	처음 보는 내용이어도 생략하지 말고 그림과 함께 읽는다.
오개념 교정	입문 설명에서 흔히 지나치게 단순화되는 문장	“어느 범위까지 맞는가”를 확인한다.
실습 / 확인 명령	허가된 실습 환경에서 상태를 관찰하는 절차	명령을 복사하기 전에 무엇을 확인하는지 먼저 말해 본다.

한 번의 웹 접속으로 보는 책 전체의 연결

브라우저에 주소를 입력하는 단순한 행동도 문자 인코딩, process와 socket, DNS, routing, ARP, Ethernet, switch, VLAN, firewall, TCP, TLS, web server, application, DB, logging, IDS/SIEM을 차례로 거친다. 각 장은 아래 흐름의 한 구간을 확대해 설명한다.

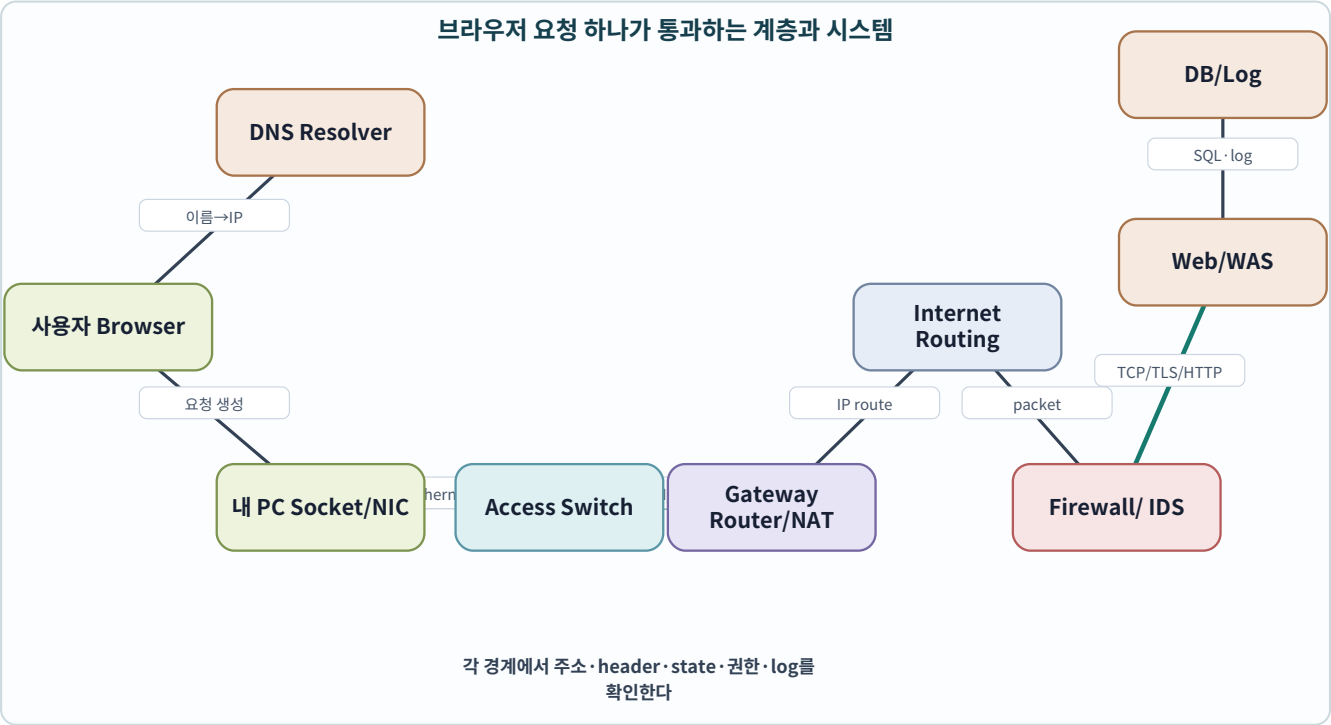


그림 0-1. 이 책은 장마다 기술을 떼어 놓지 않고 같은 요청이 어느 경계에서 어떤 형태로 바뀌는지 계속 추적한다.

권장 학습법: 그림을 다시 그릴 수 있을 때까지

1. **정의 상자**를 읽고 해당 용어의 대상과 범위를 한 문장으로 말한다.
2. **동작 그림**의 번호를 가린 뒤, 입력이 들어와 출력이 나오기까지 각 장치가 무엇을 조회하는지 설명한다.
3. **명령어**를 실행하기 전에 "이 명령은 어떤 table, process, socket, interface, rule을 보여 주는가"를 적는다.
4. **장애 시나리오**에서는 증상부터 추측하지 말고 물리 상태 → L2 → L3 → L4 → application → security policy 순으로 관찰한다.
5. **보안 시나리오**에서는 공격 문자열보다 신뢰 경계, parser, authentication, authorization, session, logging 중 무엇이 무너졌는지 찾는다.

실습 안전 원칙

Nmap, Burp Suite, packet injection, password audit, exploit 도구는 소유하거나 명시적으로 허가받은 격리된 환경에서만 사용한다. 본문의 주소는 192.0.2.0/24, 198.51.100.0/24, 203.0.113.0/24 같은 문서화용 대역을 사용하며, 실제 운영 자산을 무단으로 검사하는 절차가 아니다.

차례

쪽수는 합본 기준이다. PDF 책갈피에서도 부·장 단위로 바로 이동할 수 있다.

컴퓨터·Linux·Python·가상화의 기반 7

1장. 비트에서 프로그램 실행까지 8

2장. 프로세스·가상 메모리·함수 호출 13

3장. Linux를 층으로 이해하기 19

4장. Python을 "문법 목록"이 아니라 객체와 실행 흐름으로 이해하기 28

5장. 가상화 ·VMware·GNS3 38

같은 LAN 안에서 프레임이 움직이는 법 45

6장. Ethernet 프레임이 만들어지고 스위치까지 도착하는 과정 46

7장. 스위치는 프레임을 어떻게 배우고 전달하는가 52

8장. L2 루프가 생기고 STP가 이를 막는 전 과정 57

9장. VLAN·802.1Q·포트 시큐리티 69

IP·전송계층·라우팅·WAN의 메커니즘 87

10장. OSI 계층과 물리 통신 88

11장. IPv4 주소와 서브넷 94

12장. ARP·ICMP·DHCP·NAT 99

13장. Port·Socket·TCP·UDP와 주요 응용 프로토콜 107

14장. 라우팅 표와 경로 선택 116

15장. RIP를 통해 배우는 Distance Vector와 수렴 문제 121

16장. OSPF·EIGRP·BGP 128

17장. Serial·PPP·Frame Relay·ACL·QoS 133

서버·스토리지·웹·DB·DNS 운영 139

18장. 서버 하드웨어와 부팅 140

19장. Disk·RAID·Filesystem·LVM 144

20장. LAMP·Nginx·WAS·PHP-FPM 151

21장. DB·DBMS·SQL·Hash 156

22장. DNS·FTP·SMB·NFS·CDN 161

23장. Log·rsyslog·NTP·SNMP·RRD 168

보안 통제·패킷 경로·탐지·관제 173

24장. 보안의 기본 언어 174

25장. Linux Host 보안 181

26장. Netfilter·iptables·nftables·firewalld·UFW 187

27장. NIC에서 Socket까지 192

28장. SPAN·TAP·Inline·DAQ 197

29장. IDS·IPS·Snort·Suricata·Zeek·Security Onion 201

30장. SIEM·EDR·XDR·SOAR·MITRE ATT&CK 207

31장. Governance·Compliance·CVE·CWE·OWASP·KISA 기준 읽기 212

승인된 모의해킹·웹 보안·통합 실습 216

32장. 모의해킹 방법론과 OSINT·Scanning·취약점 분석 217

33장. Packet 분석과 공격 개념 223

34장. HTTP·Cookie·Session·Same-Origin·CORS 228

35장. SQL Injection·XSS·CSRF·접근통제 235

36장. Kali 도구와 취약 실습 플랫폼 244

37장. 통합 실습과 트러블슈팅 250

보충 용어·오개념 교정·학습 지도 255

부록 A. 네트워크 기초 용어를 빈틈없이 연결하기 256

부록 B. Linux·Python·자동화에서 자주 만나는 보충 용어 264

부록 C. 표준 문서와 프로토콜 표기 읽는 법 274

부록 D. 자주 생기는 오개념 30선 278

부록 E. 75일 필기를 개념 흐름으로 다시 배치한 학습 지도 280

부록 F. 공식 문서 중심 참고자료 282

용어 찾아보기 284

PART I

컴퓨터·Linux·Python·가상화의 기반

네트워크와 보안을 이해하려면 먼저 프로그램이 어디에서 실행되고, 운영체제가 CPU·메모리·파일·장치를 어떻게 중개하는지 알아야 한다. 이 부분은 명령어 암기보다 '명령 하나가 커널과 하드웨어에서 무엇을 일으키는가'를 설명한다.

1장. 비트에서 프로그램 실행까지: 컴퓨터는 무엇을 처리하는가

> 정보보안과 네트워크에서 패킷, 주소, 권한, 암호 키, 파일은 서로 다른 것처럼 보이지만 컴퓨터 안에서는 모두 비트의 배열이다. 이 장은 비트·바이트·진법을 출발점으로 CPU가 명령어를 읽고 실행하며, 메모리와 저장장치가 어떤 역할을 나누는지 연결한다.

비트·바이트·옥텟과 진법

필기 확인 + 보충 해설

모든 파일과 패킷이 결국 같은 재료로 표현되는 이유

함께 찾을 말: bit, byte, octet, 2진수, 16진수, binary, hexadecimal

비트·바이트·옥텟과 진법란 비트는 두 상태 중 하나를 나타내는 최소 정보 단위이고, 바이트는 현대 시스템에서 보통 8비트 묶음이며, 옥텟은 정확히 8비트임을 강조하는 표준 용어다.

0과 1은 스위치의 실제 상태를 단순화한 표현이다

비트는 물리적으로 반드시 "정확히 0볼트와 1볼트"를 뜻하지 않는다. 회로는 전압·전하·자기 상태처럼 두 범위로 안정적으로 구분할 수 있는 물리 상태를 0과 1로 해석한다. 이렇게 두 상태를 조합하면 숫자, 문자, 명령어, 그림을 같은 저장·전송 장치로 다룰 수 있다.

8비트를 한 묶음으로 보는 이유

초기 컴퓨터에는 문자 하나를 표현하기에 충분하고 하드웨어가 다루기 편한 크기가 필요했다. 오늘날 byte는 거의 항상 8bit이며, 네트워크 표준은 장비마다 byte 크기가 달랐던 역사적 혼동을 피하려고 8bit 단위를 **octet**이라고 명시한다. IPv4 주소의 192.168.1.10이 점으로 나뉜 네 칸도 각각 8bit, 즉 한 octet이다.

단위	크기	무엇을 표현하기 좋은가
bit	0 또는 1 한 자리	플래그 한 개, 참/거짓 상태
nibble	4bit	16진수 한 자리
byte/octet	8bit	0~255, ASCII 문자, IPv4 한 칸
word	CPU 설계에 따라 다름	CPU가 자연스럽게 처리하는 데이터 폭

8비트 한 옥텟의 자리값

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
128	64	32	16	8	4	2	1

그림 1-1. 각 비트가 켜져 있으면 해당 2의 거듭제곱 값을 더한다. 11111111_2는 255다.

왜 네트워크에서 16진수를 자주 쓰는가

2진수 네 자리는 16진수 한 자리와 정확히 대응한다. MAC 주소 `00:1A:2B:3C:4D:5E`, EtherType `0x0800`, 메모리 주소를 짧고 경계가 보이게 표현할 수 있다.

문자 인코딩과 ASCII

필기 확인 + 보충 해설

문자도 합의된 숫자 규칙이 있어야 전달된다

함께 찾을 말: ASCII, UTF-8, Unicode, ANSI

문자 인코딩과 ASCII란 문자 인코딩은 사람이 보는 문자와 컴퓨터가 저장하는 바이트열 사이의 대응 규칙이며, ASCII는 영문 중심의 초기 공통 대응표다.

ASCII가 해결한 문제

컴퓨터 A가 숫자 65를 문자 A로 해석하고 컴퓨터 B가 다른 문자로 해석하면 통신할 수 없다. ASCII는 영문자·숫자·제어문자에 공통 번호를 배정했다. 예를 들어 대문자 A는 65(0x41), 줄바꿈의 구성 요소인 LF는 10(0x0A)이다.

인코딩과 글꼴은 다르다

문자 인코딩은 문자를 숫자 바이트열로 바꾸는 규칙이고, 글꼴은 그 문자를 화면에 어떤 모양으로 그릴지 정한다. UTF-8 문서의 한글이 깨지는 원인은 대개 바이트를 다른 인코딩으로 해석했거나 글꼴에 해당 글리프가 없기 때문이다.

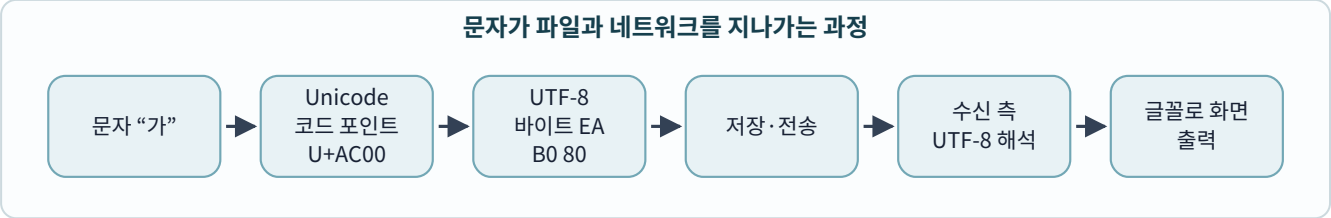


그림 1-2. 인코딩이 다르면 같은 바이트를 다른 문자로 읽게 된다.

ANSI라는 말이 모호한 이유

Windows 프로그램에서 “ANSI 형식”은 엄밀한 하나의 인코딩 이름이 아니라 현재 시스템 코드 페이지를 뜻하는 경우가 많다. 한국어 Windows에서는 과거 CP949 계열을 가리키기도 한다. 교환용 텍스트는 UTF-8처럼 인코딩을 명시하는 편이 안전하다.

CPU·명령어·마이크로프로세싱

필기 확인 + 보충 해설

고수준 코드가 작은 기계 명령으로 실행되는 과정

함께 찾을 말: CPU, Microprocessor, Fetch Decode Execute, operator, operand

CPU·명령어·마이크로프로세싱란 CPU는 메모리에서 명령어와 데이터를 가져와 산술·논리·분기·입출력 제어를 수행하는 처리 장치이며, 마이크로프로세서는 이런 CPU 기능을 하나의 집적회로에 구현한 것이다.

CPU의 일은 명령어를 반복 처리하는 것이다

CPU는 메모리에 있는 기계어 명령어를 가져오고(fetch), 해석하고(decode), 실행한 뒤(execute) 결과를 레지스터나 메모리에 기록한다. 프로그램이 복잡해 보여도 CPU 관점에서는 산술, 논리, 비교, 분기, 메모리 읽기·쓰기 같은 작은 명령의 연속이다.

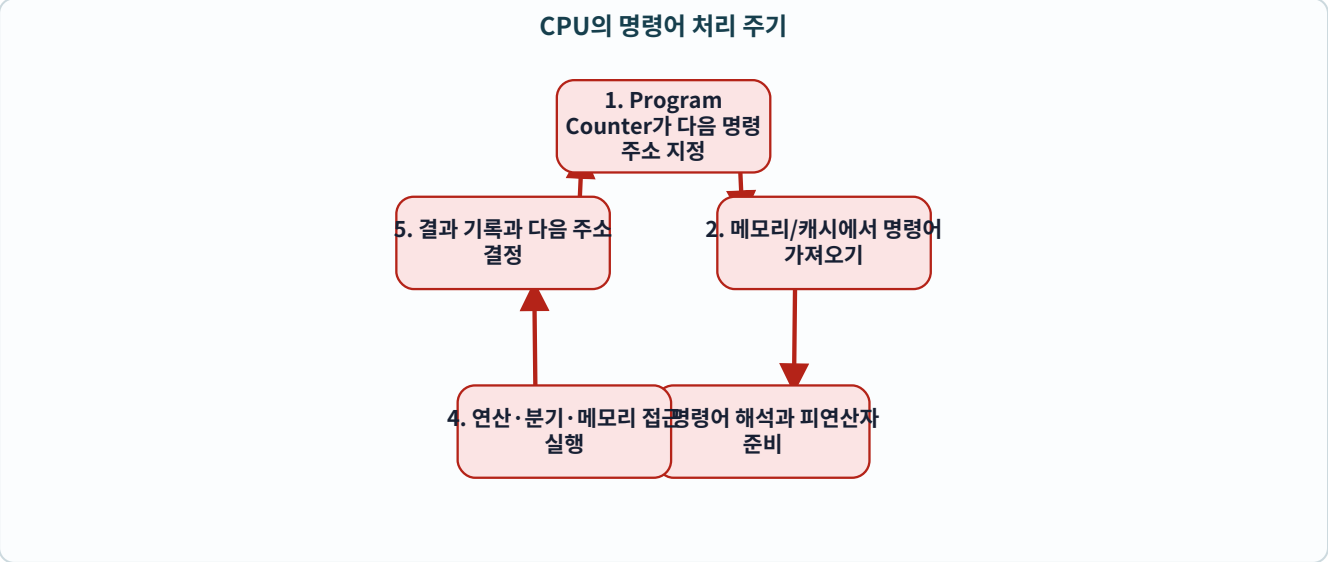


그림 1-3. 현대 CPU는 파이프라인·슈퍼스칼라·분기 예측으로 여러 단계를 겹쳐 처리하지만 논리적 흐름은 같다.

연산자와 피연산자

연산자(operator)는 무엇을 할지 나타내는 기호나 명령이고, 피연산자(operand)는 그 대상이다. a + b에서 +는 연산자, a와 b는 피연산자다. assembly의 `ADD R0, R1`도 같은 구조다.

조건문이 기계어에서 하는 일

조건문은 비교 결과에 따라 Program Counter를 다른 위치로 바꾸는 조건 분기다. “비밀번호가 맞으면 로그인 처리, 아니면 실패 처리”는 비교 명령이 상태 플래그를 설정하고 조건 점프가 어느 코드 블록을 실행할지 선택하는 형태로 내려간다.

shift가 2의 거듭제곱 연산에 쓰이는 이유

2진수에서 왼쪽으로 한 칸 밀면 각 자리값이 두 배가 된다. 양수 정수의 x << 1은 범위 내에서 x×2와 대응한다. 다만 컴파일러가 이미 최적화할 수 있고, 부호·오버플로·언어 규칙이 있으므로 사람이 무조건 곱셈을 shift로 바꾸는 것이 항상 더 빠르거나 안전한 것은 아니다.

“논리연산은 무조건 산술연산보다 무겁다”는 식의 암기는 위험하다

처리 비용은 CPU 마이크로아키텍처, 데이터 의존성, 캐시 미스, 분기 예측 실패, 벡터화 여부에 따라 달라진다. 단순 연산 한 번보다 메모리에서 데이터를 기다리는 시간이 병목인 경우가 흔하다.

메모리와 저장장치

필기 확인 + 보충 해설

"파일을 실행한다"는 말 안에서 일어나는 데이터 이동

함께 찾을 말: RAM, Storage, Memory allocation, cache

메모리와 저장장치란 메모리는 CPU가 실행 중인 명령과 데이터를 빠르게 읽고 쓰는 작업 공간이고, 저장장치는 전원이 꺼져도 파일을 보존하는 장치다.

레지스터·캐시·RAM·저장장치의 역할

CPU에 가까울수록 빠르고 작고 비싸다. 레지스터는 현재 연산에 즉시 쓰는 값을, 캐시는 최근·인접 데이터를, RAM은 실행 중인 프로그램과 데이터를, SSD/HDD는 전원이 꺼져도 남길 파일을 저장한다. 프로그램 실행은 저장장치의 실행 파일 일부를 RAM의 가상 주소 공간으로 적재하는 것에서 시작한다.

계층	전원 종료 후	주 용도	상대적 특성
레지스터	사라짐	현재 명령의 값·주소	가장 빠르고 매우 작음
CPU 캐시	사라짐	최근 사용한 명령·데이터	매우 빠름
RAM	사라짐	실행 중 코드·힙·스택·버퍼	빠르고 용량 큼
SSD/HDD	유지	파일·OS·DB	느리지만 영구 저장

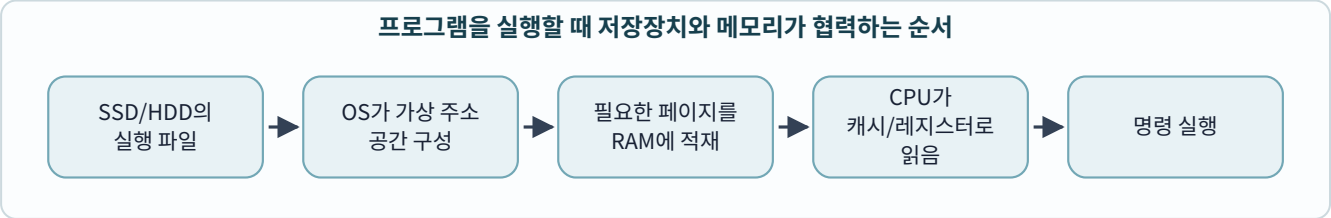


그림 1-4. "메모리에 올린다"는 말은 파일의 필요한 부분을 프로세스 주소 공간에 매핑하고 물리 RAM 페이지와 연결하는 과정이다.

메모리를 할당한다는 뜻

프로그램이 "메모리 1MB를 할당한다"는 것은 OS나 런타임에 주소 범위를 요청해 그 범위를 자신이 사용할 수 있도록 예약·매핑받는다는 뜻이다. 처음부터 모든 물리 RAM이 즉시 채워지는 것은 아니며, 페이지를 실제 접근할 때 물리 메모리가 배정되는 demand paging이 사용될 수 있다.

2장. 프로세스·가상 메모리·함수 호출: 실행 중 프로그램의 내부

> 실행 파일은 저장된 명령어 묶음이고, 프로세스는 운영체제가 그 프로그램을 실행하도록 메모리·CPU 시간·파일 핸들 같은 자원을 붙여 관리하는 단위다. 이 장에서는 code, data, BSS, heap, stack을 단어로만 외우지 않고 변수와 함수 호출이 어디에 놓이는지 추적한다.

프로그램·프로세스·PID

필기 확인 + 보충 해설

파일이 실행 상태가 될 때 붙는 운영체제 정보

함께 찾을 말: Process, Program, PID, PPID

프로그램·프로세스·PID란 프로그램은 저장장치에 있는 명령과 자원의 정적 파일이고, 프로세스는 운영체제가 그 프로그램을 실행하며 자원과 상태를 관리하는 실행 인스턴스이며, PID는 그 인스턴스의 식별 번호다.

프로그램과 프로세스의 차이

프로그램은 디스크에 있는 실행 파일과 관련 자원의 정적인 묶음이다. 프로세스는 그 프로그램이 실행되면서 PID, 가상 주소 공간, 열린 파일, 사용자 권한, CPU 상태를 가진 동적인 실행 인스턴스다. 같은 프로그램을 두 번 실행하면 보통 PID와 메모리가 다른 두 프로세스가 생긴다.

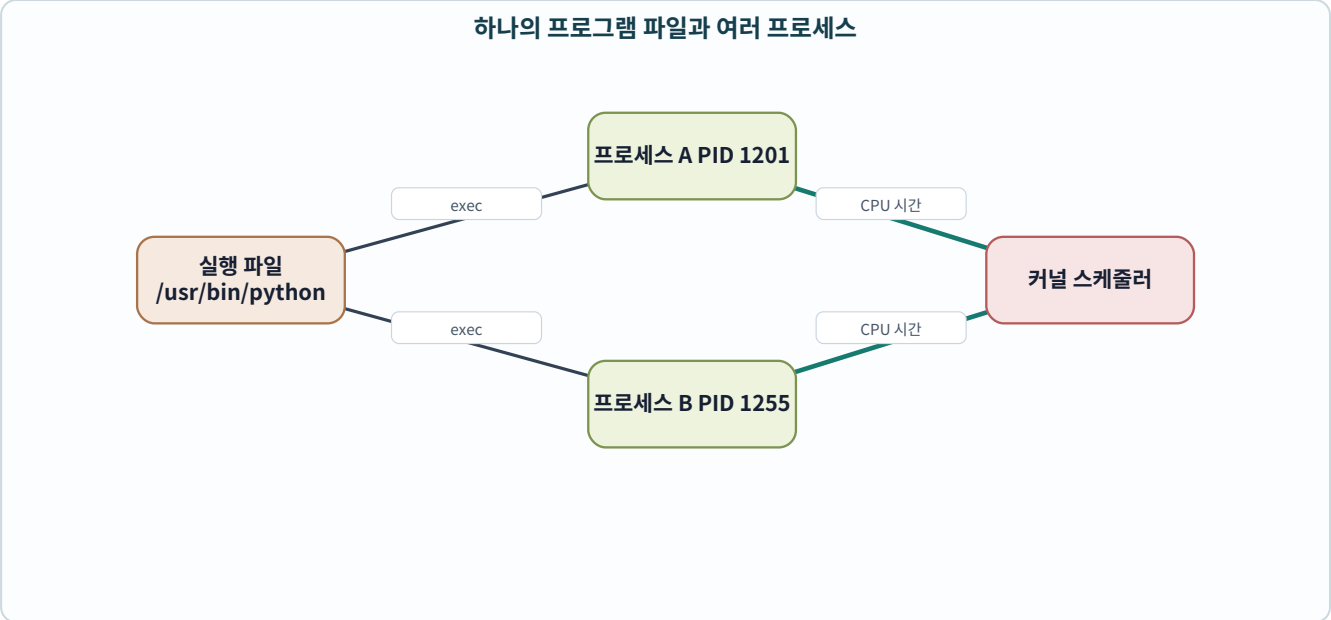


그림 2-1. 프로세스마다 별도 가상 주소 공간과 실행 상태를 갖고, 스케줄러가 CPU 시간을 나눈다.

PID와 부모·자식 관계

PID(Process ID)는 실행 중 프로세스를 구분하는 번호다. Linux에서 새 프로세스는 흔히 기존 프로세스가 fork로 복제된 뒤 exec로 다른 프로그램을 적재하는 방식으로 만들어진다. PPID는 부모 프로세스 번호다. systemd가 많은 서비스의 조상 역할을 한다.

프로세스 관계와 상태 확인

```
ps -ef
ps -o pid,ppid,user,stat,cmd -p 1234
cat /proc/1234/status
pstree -p
```

프로세스 메모리 구조

필기 확인 + OS/ABI 보강

변수와 함수가 “메모리 어디에 있는가”를 설명하는 지도

함께 찾을 말: Code, Text, Data, BSS, Heap, Stack, stack frame

프로세스 메모리 구조란 프로세스의 가상 주소 공간을 코드, 정적 데이터, 동적 할당 영역, 함수 호출 영역 등 역할별로 나누어 관리하는 구조다.

프로세스 주소 공간의 대표 구역

구역	주로 들어가는 것	성질
Text/Code	기계어 명령	대개 읽기·실행 가능, 쓰기 금지
Read-only data	문자열 리터럴 등	읽기 전용
Data	초기값이 있는 전역·static 변수	실행 파일에 초기값 포함
BSS	0으로 초기화되거나 초기값 없는 전역·static	파일에는 크기 정보 중심
Heap	동적 할당 객체	필요에 따라 확장·해제
Stack	함수 호출 프레임, 지역 변수, 복귀 주소	호출과 반환에 따라 증감



그림 2-2. 실제 배치와 성장 방향은 OS·아키텍처에 따라 달라질 수 있지만 역할 구분을 이해하는 모델이다.

함수를 호출할 때 stack에서 일어나는 일

함수 호출 시 CPU와 컴파일러의 호출 규약에 따라 인자 일부, 복귀 주소, 이전 프레임 포인터, 저장해야 할 레지스터, 지역 변수를 위한 공간이 stack frame에 배치된다. 함수가 반환되면 frame을 되돌리고 복귀 주소로 실행을 이어간다. 재귀 호출이 너무 깊으면 stack 공간을 소진해 stack overflow가 난다.

Heap과 메모리 누수

Heap은 실행 중 크기를 미리 알 수 없는 객체나 오래 살아야 하는 데이터를 위한 영역이다. C처럼 명시적 해제가 필요한 언어에서 더 이상 접근할 수 없는 메모리를 해제하지 않으면 누수가 발생한다. Python은 참조 카운트와 순환 가비지 컬렉터가 객체 생명주기를 관리하지만, 파일·소켓 같은 외부 자원은 여전히 명시적 close나 context manager가 중요하다.

stack에는 값, heap에는 객체라는 단순 공식은 예외가 많다

언어 런타임, 최적화, escape analysis에 따라 실제 배치는 달라진다. Python 변수 이름은 대체로 객체에 대한 참조를 가리키고, Python 객체는 런타임이 관리하는 heap에 존재한다고 이해하는 편이 낫다.

가상 메모리와 Swap

필기 확인 + OS 보강

격리와 유연성을 얻는 대신 주소 변환이 필요하다

함께 찾을 말: Virtual Memory, Page Table, MMU, Page Fault, swap

가상 메모리와 Swap란 가상 메모리는 프로세스가 독립된 연속 주소 공간을 쓰는 것처럼 보이게 하고 이를 실제 RAM·파일과 매핑하는 메모리 관리 방식이며, swap은 메모리 페이지를 저장장치에 임시 대피시키는 공간이다.

가상 주소를 쓰는 이유

각 프로세스가 자신만의 0번지부터 시작하는 연속된 주소 공간을 가진 것처럼 보이게 하면, 다른 프로세스의 메모리와 격리하고 실제 RAM의 흩어진 페이지를 유연하게 배치할 수 있다. CPU의 MMU와 OS page table이 가상 주소를 물리 주소로 변환하며 접근 권한도 검사한다.

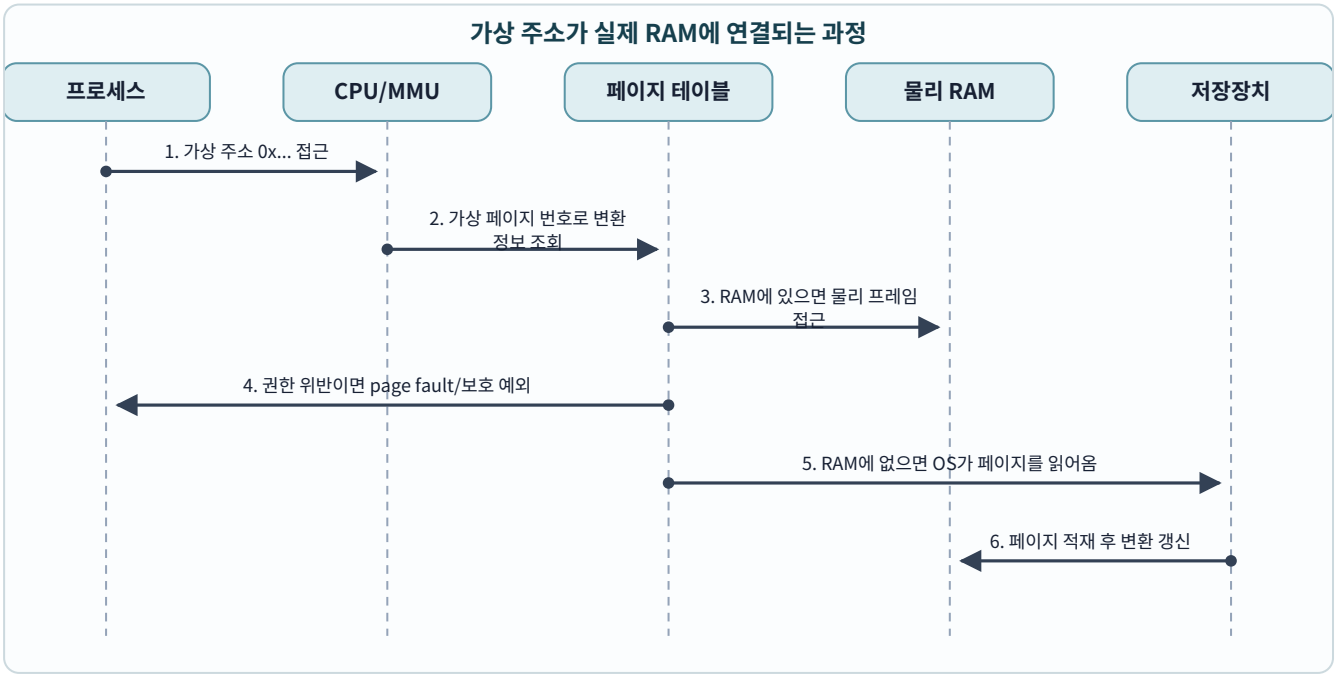


그림 2-3. Page fault는 항상 오류가 아니라, 아직 적재되지 않은 정상 페이지를 불러오기 위한 이벤트일 수도 있다.

Swap의 역할과 한계

메모리 압박이 있을 때 OS는 당장 쓰지 않는 메모리 페이지를 swap 영역으로 내보내 RAM을 확보할 수 있다. 다시 접근하면 저장장치에서 읽어와야 하므로 매우 느리다. swap은 RAM을 완전히 대체하는 확장 메모리가 아니라 급격한 메모리 부족을 완충하는 수단이다. 지속적인 swap in/out은 thrashing과 성능 붕괴를 일으킨다.

메모리와 swap 관찰

```
free -h
swapon --show
vmstat 1
cat /proc/meminfo
pmap -x <PID>
```

포인터·참조·인자 전달

필기 확인 + 보충 해설

Python 변수와 C 포인터를 같은 말로 혼동하지 않기

함께 찾을 말: Pointer, Reference, Call by Value, Call by Reference, object sharing

포인터·참조·인자 전달란 포인터는 메모리 주소를 저장하고 역참조할 수 있는 값이며, 참조는 객체를 간접적으로 가리키는 연결을 뜻한다. 인자 전달 방식은 함수가 무엇의 복사본 또는 별칭을 받는지 설명한다.

포인터란 주소를 값으로 다루는 변수다

C의 포인터는 어떤 데이터가 있는 메모리 주소를 저장한다. `int *p = &x;`에서 &x는 x의 주소, p는 그 주소를 저장하며, *p는 그 주소에 있는 값을 역참조한다. 잘못된 주소·해제된 주소를 역참조하면 crash나 취약점으로 이어질 수 있다.

Call by value와 reference를 구분하는 기준

Call by value는 인자의 값을 복사해 전달하므로 함수가 매개변수 변수 자체를 바꿔도 호출자 변수의 바인딩은 바뀌지 않는다. Call by reference는 호출자 변수 자체에 대한 별칭을 전달한다. C는 값 전달만 제공하지만 포인터 값을 전달해 간접적으로 호출자 데이터를 수정한다. Python도 객체 참조 값을 전달하는 방식이어서 “call by object sharing” 또는 “call by assignment”라고 설명한다.

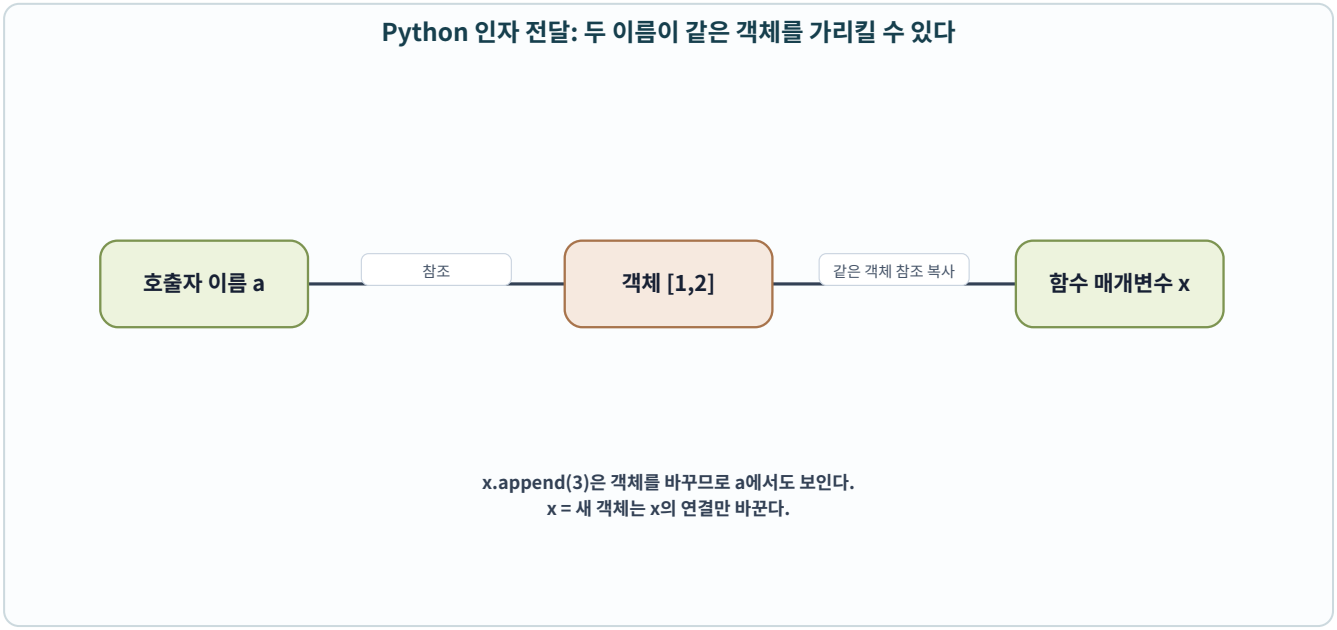


그림 2-4. Python은 객체 자체를 통째로 복사하지 않고 객체를 가리키는 참조가 매개변수 이름에 바인딩된다.

보안과 포인터

경계 검사가 잘못된 포인터 연산은 buffer overflow, use-after-free, arbitrary memory write 같은 취약점의 토대가 된다. Python은 직접 주소 연산을 숨겨 이런 오류를 줄이지만, C 확장 모듈과 런타임 아래에는 여전히 메모리 안전 문제가 존재할 수 있다.

3장. Linux를 층으로 이해하기: 커널·셸·파일·권한·서비스

> 터미널에서 명령어가 실행되는 경로는 “키보드 → 터미널 → 셸 → 시스템 콜 → 커널 → 파일 시스템·드라이버”다. 이 층을 구분하면 console, terminal, shell, bash, sudo, service, daemon 같은 말이 서로 어떻게 다른지 보인다.

Linux 커널·셸·터미널

필기 확인 + 보충 해설

화면에 보이는 프롬프트와 실제 OS 핵심을 분리해서 보기

함께 찾을 말: Linux, Unix, Kernel, Shell, Terminal, Console, CLI, GUI

Linux 커널·셸·터미널란 커널은 CPU·메모리·파일·장치를 통제하는 운영체제 핵심이고, 셸은 사용자의 명령을 해석해 프로그램을 실행하는 사용자 공간 프로그램이며, 터미널은 문자 입출력 통로다.

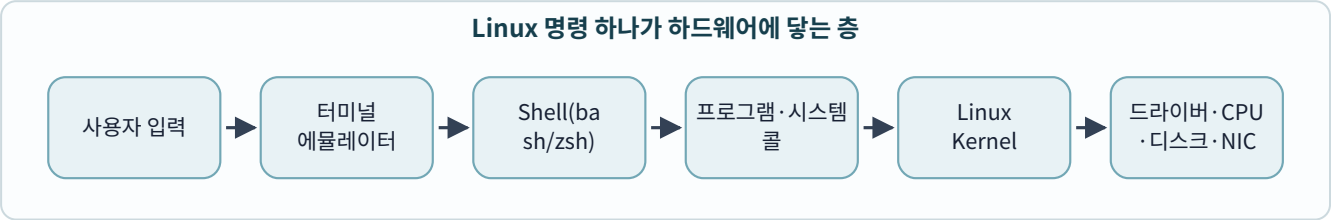


그림 3-1. 셸은 명령을 해석하는 사용자 공간 프로그램이고, 커널은 권한과 자원을 통제한다.

Unix와 Linux

Unix는 1970년대부터 발전한 운영체제 계열과 설계 전통을 가리킨다. Linux는 Unix와 호환되는 철학과 인터페이스를 많이 따르지만 원래 Unix 코드에서 파생된 단일 제품이라는 뜻은 아니다. 배포판은 Linux kernel에 GNU 도구, 패키지 관리자, 설치기, 데스크톱 등을 조합한 운영체제 묶음이다.

Console·Terminal·Shell

Console

원래 장비를 직접 제어하는 물리 입출력 장치나 관리 접속을 뜻한다. 네트워크 장비의 console port는 네트워크가 없어도 로컬 관리할 수 있는 경로다.

Terminal

문자를 입력하고 출력하는 끝단 장치 또는 그 기능을 흉내 내는 프로그램이다. GNOME Terminal, Windows Terminal은 terminal emulator다.

Shell

명령문을 해석하고 프로그램 실행, 변수, 파이프, 리다이렉션, 스크립트를 제공하는 프로그램이다. bash, zsh, sh가 예다.

CLI와 GUI

CLI는 문자 명령 중심의 인터페이스, GUI는 창·아이콘 중심의 인터페이스다. CLI가 “저수준”, GUI가 “고수준”이라는 뜻은 아니다. 자동화와 원격 관리에서는 명령이 기록·재현되기 쉬워 CLI가 널리 쓰인다.

Linux 파일시스템 계층과 “모든 것은 파일”

필기 확인 + 보충 해설

일반 파일과 커널 인터페이스 파일을 구분하기

함께 찾을 말: FHS, file descriptor, /proc, /sys, /dev, /etc, /var

Linux 파일시스템 계층과 “모든 것은 파일”란 Linux는 서로 다른 커널 자원을 file descriptor와 read/write 같은 공통 인터페이스로 다룰 수 있게 추상화하며, 디렉터리 계층은 설정·로그·장치·프로세스 정보의 전통적 위치를 정한다.

“모든 것은 파일이다”의 정확한 의미

Linux가 모든 대상을 디스크의 일반 파일로 저장한다는 뜻이 아니다. 프로세스는 일반 파일, 디바이스, pipe, socket 등을 **file descriptor**라는 공통 핸들로 다루고 read/write/close 같은 공통 인터페이스를 사용할 수 있다. 종류별 세부 동작은 커널 객체와 드라이버가 구현한다.

경로	전통적 역할	초보자가 확인할 점
/etc	시스템·서비스 설정	프로그램별 conf, conf.d
/var	실행 중 변하는 데이터	/var/log, spool, cache
/proc	커널이 보여주는 프로세스·상태 가상 파일	실제 디스크 파일이 아님
/sys	장치·드라이버·커널 객체 가상 계층	sysfs
/dev	장치 노드와 특수 파일	null, zero, random, disk
/run	부팅 후 실행 상태·PID·socket	재부팅 시 초기화
/usr	사용자 공간 프로그램·라이브러리·공유 데이터	/usr/bin, /usr/lib
/srv	서비스가 제공하는 데이터용 관례	사이트 정책에 따라 사용
/opt	추가 애플리케이션 묶음용 관례	벤더 소프트웨어

/dev/null·/dev/zero·/dev/random

/dev/null

쓴 데이터는 버리고, 읽으면 즉시 EOF를 반환하는 sink다.

/dev/zero

읽을 때 0바이트를 계속 제공한다. 빈 이미지나 초기화 테스트에 쓸 수 있다.

/dev/random·urandom

커널 난수 생성기 인터페이스다. 현대 Linux의 정확한 blocking 특성은 커널 버전에 따라 설명이 단순 교재와 다를 수 있으므로 암호용 코드는 언어의 안전한 CSPRNG API를 쓰는 편이 낫다.

가상 파일과 file descriptor 관찰

```
ls -l /dev/null /dev/zero /dev/random
cat /proc/cpuinfo | head
cat /proc/<PID>/fd/0
readlink /proc/<PID>/fd/1
```

Linux 권한·chmod·umask·Sticky bit

필기 확인 + 보충 해설

숫자를 외우기보다 "누가 어떤 객체에 어떤 작업을 하는가"를 판정한다

함께 찾을 말: rwx, chmod, chown, umask, sticky bit, permission

Linux 권한·chmod·umask·Sticky bit란 Linux의 전통 권한 모델은 소유자·그룹·그 밖의 사용자별로 읽기·쓰기·실행 비트를 두며, chmod는 이를 변경하고 umask는 새 객체의 기본 허용 권한을 제한한다.

rwx의 대상은 user·group·other 세 묶음이다

권한 표시는 소유자(user), 소유 그룹(group), 그 밖의 사용자(other)에 대해 read, write, execute 비트를 따로 둔다. 파일에서 r은 내용 읽기, w는 내용 수정, x는 실행 권한이다. 디렉터리에서는 r이 이름 목록 읽기, w가 항목 생성·삭제, x가 경로 통과와 내부 inode 접근에 가깝다.



그림 3-2. 예: -rwxr-x--- 는 소유자 rwx, 그룹 r-x, 그 외 ---다.

숫자 chmod는 비트의 합이다

r=4, w=2, x=1로 보고 각 묶음에서 합한다. 750은 user 7(rwx), group 5(r-x), other 0(---)다. chmod 777은 문제를 해결하는 만능 명령이 아니라 모든 로컬 사용자가 수정·실행할 수 있게 만들어 공격면을 크게 늘릴 수 있다.

umask는 새 파일에서 빼는 기본 제한이다

프로그램이 요청한 기본 권한에서 umask가 막는 비트를 제거한다. 일반 파일은 실행 비트를 기본으로 요청하지 않는 경우가 많아 666에서, 디렉터리는 777에서 계산하는 설명이 흔하다. 다만 실제 모드는 프로그램 요청값, ACL, 마운트 옵션의 영향도 받는다.

Sticky bit

공용 쓰기 디렉터리에 sticky bit가 있으면 사용자가 디렉터리에 쓸 수 있어도 다른 사용자가 소유한 파일을 마음대로 삭제·이름 변경하지 못하게 제한한다. /tmp의 drwxrwxrwt가 대표적이다.

권한과 소유권 확인

```
id
ls -ld /tmp /var/www /path/to/file
stat /path/to/file
chmod 750 script.sh
chown webadmin:www-data /srv/site
umask
getfacl /path/to/file
```

파일을 삭제하는 권한은 파일보다 부모 디렉터리가 좌우한다

삭제는 디렉터리 항목에서 이름을 제거하는 작업이므로 부모 디렉터리의 write+execute 권한이 핵심이다. 파일 자체가 읽기 전용이어도 디렉터리 권한이 있으면 삭제할 수 있는 이유다.

셸 파이프·리다이렉션·종료 상태

필기 확인 + 보충 해설

명령 여러 개가 파일과 프로세스로 연결되는 방식

함께 찾을 말: pipe, |, &&, redirection, exit code, return 0, SIGINT, Ctrl+C

셸 파이프·리다이렉션·종료 상태란 파이프는 한 프로세스의 출력 스트림을 다른 프로세스의 입력 스트림에 연결하고, 리다이렉션은 표준 입출력의 목적지를 바꾸며, 종료 상태는 명령 성공 여부를 숫자로 전달한다.

파이프와 리다이렉션

¦는 왼쪽 프로그램의 표준 출력을 커널 pipe를 통해 오른쪽 프로그램의 표준 입력으로 연결한다. >는 출력을 파일로 새로 쓰고, >>는 끝에 추가한다. 2>는 표준 오류를 따로 보낸다. 파이프는 단순 문자열 치환이 아니라 두 프로세스 사이의 바이트 스트림 IPC다.

&&와 ;의 차이

cmd1 && cmd2는 cmd1의 종료 상태가 성공(관례상 0)일 때만 cmd2를 실행한다. ;는 앞 명령 성공 여부와 관계없이 다음 명령을 실행한다. ¦¦는 앞 명령이 실패했을 때 다음 명령을 실행한다.

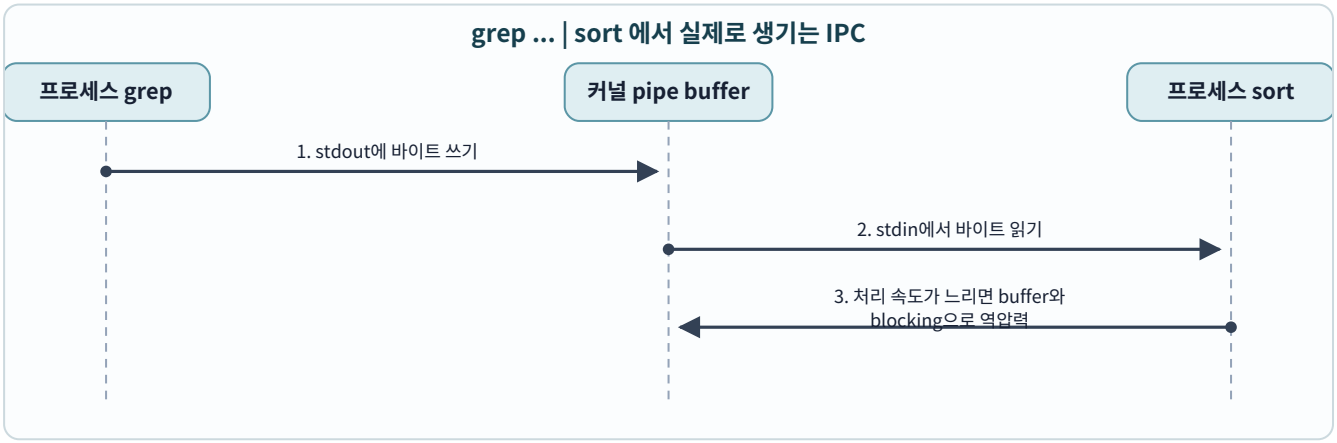


그림 3-3. 셸이 pipe를 만들고 양쪽 프로세스의 file descriptor 1과 0을 각각 연결한다.

셸 연산자와 pipe

```
# 성공한 경우에만 서비스 재시작
nginx -t && sudo systemctl reload nginx

# 로그에서 ERROR를 찾아 개수 계산
cat app.log | grep 'ERROR' | wc -l
# cat은 불필요하므로 보통 다음이 더 간단하다
grep 'ERROR' app.log | wc -l

# stdout과 stderr 분리
command >result.txt 2>error.txt
```

Ctrl+C의 의미

터미널 드라이버가 현재 foreground process group에 SIGINT 신호를 보낸다. 프로그램은 기본 동작대로 종료되거나 신호 처리기를 통해 정리 후 종료할 수 있다. 전원 차단과 같은 강제 중단이 아니므로 프로그램이 신호를 무시하면 종료되지 않을 수도 있다.

systemd·서비스·daemon

필기 확인 + 보충 해설

프로세스를 “실행만” 하는 것과 운영 상태로 관리하는 것의 차이

함께 찾을 말: init, systemd, service, daemon, unit, systemctl

---

systemd·서비스·daemon란 daemon은 백그라운드에서 지속적으로 동작하는 프로세스이고, systemd는 unit 정의를 바탕으로 서비스의 시작·종료·의존성·재시작·로그를 관리하는 init 시스템이다.

Service와 daemon

daemon은 백그라운드에서 지속적으로 요청이나 사건을 기다리는 프로세스를 말한다. service는 그 daemon을 시작·중지·의존성 관리·상태 확인하는 운영 단위까지 포함하는 말로 쓰인다. Linux의 systemd에서는 unit 파일이 실행 명령, 재시작 정책, 순서와 의존성을 선언한다.

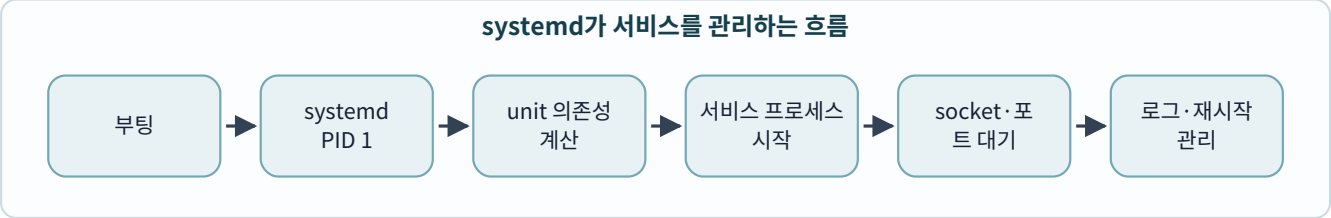


그림 3-4. enable은 부팅 시 시작 관계를 만들고, start는 지금 실행한다. 둘은 같은 동작이 아니다.

systemd 서비스 확인

```
systemctl status sshd
sudo systemctl enable --now sshd
sudo systemctl restart nginx
journalctl -u nginx --since today
systemctl cat nginx
```

init과 systemd

init은 전통적으로 PID 1에서 사용자 공간을 시작하고 서비스를 관리하는 역할을 가리킨다. systemd는 현재 많은 Linux 배포판에서 init 시스템 역할을 하며 병렬 시작, socket activation, journal, cgroup 통합 등을 제공한다.

프로그램 전용 사용자

웹 서버를 root로 계속 실행하면 취약점 하나가 전체 시스템 권한으로 이어질 수 있다. 그래서 설치 시 로그인 셸이 `/sbin/nologin`인 system user/group을 만들고, 필요한 디렉터리와 포트·capability만 허용하는 최소 권한 설계를 한다.

패키지 관리자·Repository·upstream/downstream

필기 확인 + 보충 해설

프로그램 한 개가 배포판에 안전하게 들어오는 공급 흐름

함께 찾을 말: repo, repository, dnf, apt, wget, curl, upstream, downstream

패키지 관리자·Repository·upstream/downstream란 패키지 관리자는 신뢰된 저장소에서 프로그램과 의존성을 설치·업데이트·검증하고, upstream은 원본 개발 흐름, downstream은 이를 받아 배포·운영하는 흐름을 뜻한다.

패키지와 저장소

패키지는 프로그램 파일, 라이브러리, 설정 템플릿, 메타데이터, 의존성을 배포 가능한 단위로 묶은 것이다. repository는 서명된 패키지와 목록을 제공하는 저장소다. dnf와 apt는 저장소 메타데이터를 읽고 필요한 의존성을 함께 계산한다.

upstream과 downstream

upstream은 원본 프로젝트나 변경이 흘러오는 상류, downstream은 이를 받아 배포판·제품·사용 환경에 적용하는 하류를 뜻한다. 배포판 패키지는 upstream 소스에 보안 패치와 빌드 설정을 적용한 downstream 결과일 수 있다.

wget·curl

wget은 URL에서 파일을 내려받고 재귀 다운로드하는 용도로 널리 쓰인다. curl은 HTTP뿐 아니라 다양한 프로토콜의 요청·응답을 세밀하게 시험하고 헤더·본문·인증서를 관찰하는 데 강하다. 다운로드한 shell script를 곧바로 root로 pipe 실행하는 습관은 공급망 위험이 크다.

패키지와 HTTP 도구

```
# Debian/Ubuntu
sudo apt update
apt show nginx
apt-cache policy nginx

# RHEL/Rocky 계열
sudo dnf repolist
sudo dnf info nginx

# HTTP 응답 헤더만 확인
curl -I https://example.com
# 파일을 이름을 지정해 저장
wget -O package.tar.gz https://example.com/package.tar.gz
```

4장. Python을 “문법 목록”이 아니라 객체와 실행 흐름으로 이해하기

> Python의 핵심은 “변수 상자에 값이 들어간다”보다 이름이 객체를 가리키고, 객체가 타입과 상태를 가진다는 데 있다. 이 관점으로 mutable/immutable, 함수 인자, 메서드, scope, closure, 파일 close까지 연결한다.

Python 이름·객체·타입

필기 확인 + 보충 해설

변수 상자 비유의 한계를 넘어서는 핵심 모델

함께 찾을 말: variable, object, data type, reference, dynamic typing

Python 이름·객체·타입란 Python에서 변수는 객체를 가리키는 이름이고, 객체는 자신의 타입·값·동작을 가지며, 대입은 이름과 객체 사이의 바인딩을 만든다.

이름·객체·타입

`x = 10`은 10이라는 int 객체를 만들거나 재사용하고 이름 x를 그 객체에 바인딩한다. x 자체에 C의 고정 크기 int 칸이 생긴다고 보는 모델은 Python의 재할당과 동적 타입을 설명하기 어렵다. `x = "ten"`을 실행하면 이름 x가 다른 str 객체를 가리킨다.

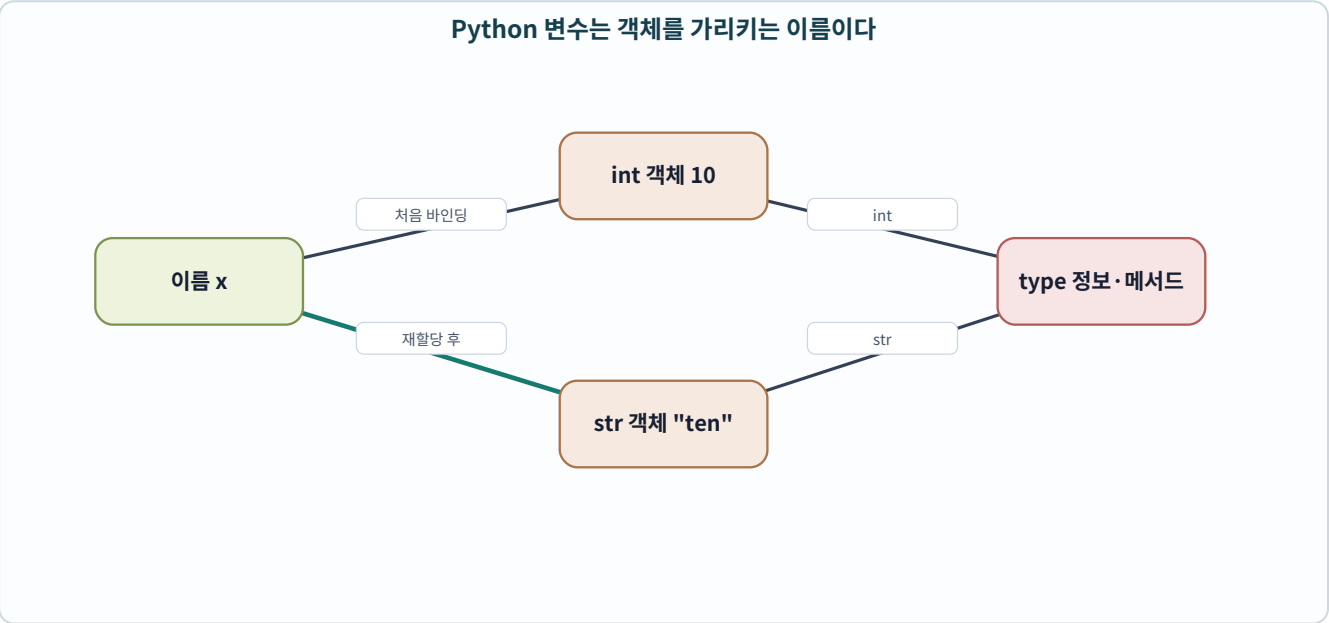


그림 4-1. 재할당은 기존 객체 내부를 반드시 바꾸는 것이 아니라 이름의 연결을 바꾼다.

“모든 것은 객체다”의 의미

숫자, 문자열, 함수, 클래스도 타입과 identity를 가진 객체로 다룰 수 있다는 뜻이다. 함수를 변수에 담고 인자로 넘길 수 있는 1급 객체 특성도 여기에서 나온다. 구현 세부는 CPython, PyPy 등 인터프리터에 따라 다를 수 있다.

참조형이라는 말의 주의점

Python의 모든 이름이 객체 참조를 사용한다. 흔히 list/dict를 “참조형”, int/str를 “값형”이라고 부르지만 더 정확한 차이는 객체가 생성 후 내부 상태를 바꿀 수 있는지인 mutable/immutable이다.

Python 자료형과 가변성

필기 확인 + 보충 해설

메서드 실행 후 원본이 바뀌는지 판단하는 기준

함께 찾을 말: list, tuple, str, mutable, immutable, formatted string, f-string

Python 자료형과 가변성란 자료형은 객체가 표현할 값과 허용 연산을 정의하고, 가변성은 객체 생성 후 그 객체의 내부 상태를 바꿀 수 있는지를 뜻한다.

Mutable과 Immutable

분류	예	같은 객체 내부 변경
immutable	int, float, bool, str, tuple(원소 구성)	불가. 연산 결과는 새 객체 바인딩
mutable	list, dict, set, 사용자 정의 객체 다수	메서드로 내부 상태 변경 가능

list와 tuple

list는 순서 있는 가변 컨테이너라 append, extend, insert, pop, sort로 내용이 바뀐다. tuple은 순서 있는 불변 컨테이너라 생성 후 원소 바인딩을 바꿀 수 없다. tuple 안에 mutable 객체가 있으면 그 내부까지 자동으로 불변이 되는 것은 아니다.

문자열 연산의 특징

str은 immutable이다. `s += part`를 반복하면 새 문자열 생성 비용이 커질 수 있어 많은 조각은 list에 모아 `"".join(parts)`하는 패턴이 효율적이다. format string은 값을 문자열에 안전하게 표현하는 편의 기능이지 SQL 쿼리나 shell 명령을 안전하게 만드는 보안 경계가 아니다.

객체 변경과 문자열 형식화

```
items = ["ARP", "TCP"]
items.append("HTTP")      # 같은 list 객체가 변경됨

name = "network"
name = name.upper()        # 새 str 결과에 다시 바인딩

host = "10.0.0.1"
port = 443
message = f"connect to {host}:{port}"
```

복사와 별칭

`b = a`는 list를 복제하지 않고 같은 객체에 두 이름을 붙인다. 독립 복사가 필요하면 얕은 복사와 깊은 복사의 차이를 이해해야 한다.

함수·메서드·인자·1급 객체

필기 확인 + 보충 해설

코드 블록도 값처럼 전달할 수 있는 Python

함께 찾을 말: function, method, parameter, argument, args, kwargs, lambda, first-class object

함수·메서드·인자·1급 객체란 함수는 호출 가능한 코드 객체이고, 메서드는 객체나 클래스에 결합된 함수이며, parameter는 정의의 입력 자리, argument는 호출 시 전달되는 실제 값이다.

함수와 메서드

함수는 입력을 받아 작업하고 결과를 반환하도록 이름 붙인 코드 객체다. 메서드는 클래스/객체의 속성으로 접근하며 호출 시 특정 객체나 클래스가 암묵적 첫 인자로 연결되는 함수다. `len(items)`는 built-in function, `items.append(x)`는 list 객체의 method다.

Parameter와 Argument

parameter는 함수 정의에 적힌 입력 자리 이름이고, argument는 호출할 때 실제로 전달하는 값이다. positional argument는 위치로, keyword argument는 이름으로 parameter에 대응한다. `*args`는 남는 위치 인자, `**kwargs`는 남는 키워드 인자를 모은다.

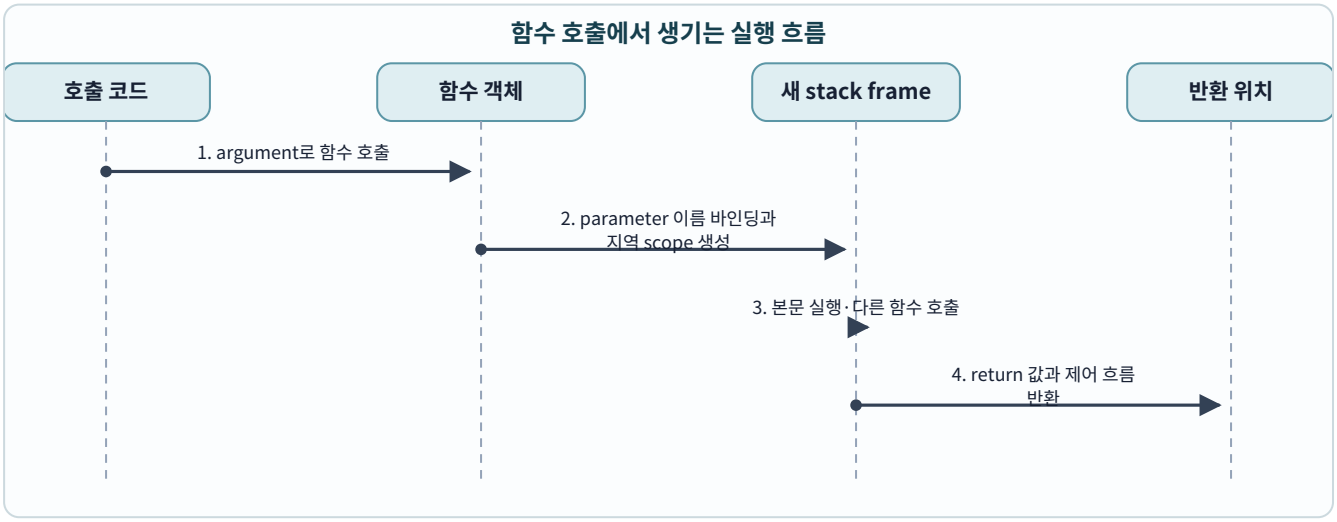


그림 4-2. return이 없으면 Python 함수는 None을 반환한다. 프로세스 종료 코드와 함수 return은 다른 층의 개념이다.

1급 객체와 lambda

1급 객체는 변수에 저장하고, 함수 인자로 전달하고, 함수 결과로 반환할 수 있는 객체다. Python 함수가 1급 객체이므로 callback, decorator, closure를 만들 수 있다. lambda는 작은 익명 함수 표현식이며 복잡한 로직을 억지로 한 줄에 넣는 도구가 아니다.

함수 객체와 closure

```
def apply(value, fn):
    return fn(value)

result = apply(5, lambda x: x * 2)

def make_prefix(prefix):
    def add_prefix(message):
        return f"{prefix}: {message}"
    return add_prefix

warn = make_prefix("WARN")
print(warn("link down"))
```

변수
Scope·LEGB·Closure

필기 확인 + 보충 해설

같은 이름이 있어도 어느 값을 읽는지 결정하는 규칙

함께 찾을 말: scope, global, nonlocal, LEGB, closure

변수 Scope·LEGB·Closure란 scope는 이름이 유효하고 검색되는 범위이며, Python은 LEGB 순서로 이름을 찾고, closure는 내부 함수가 바깥 함수의 환경을 유지하는 구조다.

LEGB 검색 순서

이름을 읽을 때 현재 함수의 Local, 바깥 함수의 Enclosing, 모듈의 Global, builtins 순으로 찾는다. 대입은 기본적으로 현재 local 이름을 만든다. 바깥 이름에 재바인딩하려면 global 또는 nonlocal 선언이 필요하지만, 남용하면 의존성이 숨겨져 테스트와 동시성이 어려워진다.



그림 4-3. 가까운 scope에서 이름을 찾으면 뒤 단계는 보지 않는다.

Closure

내부 함수가 바깥 함수 호출이 끝난 뒤에도 바깥 scope의 변수를 기억하는 구조다. 상태를 캡슐화하거나 callback을 구성할 수 있다. 캡처되는 것은 “그 순간 값의 완전한 복사본”이라고 단정할 수 없고 변수 cell에 대한 참조라는 점 때문에 loop와 lambda에서 late binding 주의가 필요하다.

global은 편하지만 결합도를 높인다

여러 함수가 하나의 global mutable 상태를 수정하면 실행 순서, 테스트 격리, thread 안전성을 추론하기 어렵다. 설정 객체나 명시적 인자로 의존성을 넘기는 설계가 대개 더 분명하다.

조건문·반복문·종료 조건

필기 확인 + 보충 해설

코드 분기와 반복이 실제 제어 흐름을 바꾸는 방식

함께 찾을 말: if, for, while, range, nested code, exit condition, infinite loop, pass, continue, break, unreachable code

조건문·반복문·종료 조건란 조건문은 평가 결과에 따라 실행 경로를 나누고, 반복문은 iterable이나 조건에 따라 블록을 재실행하며, 종료 조건은 반복이 끝나는 상태를 정의한다.

반복문의 본질

반복문은 종료 조건이 만족될 때까지 코드 블록을 다시 실행한다. C의 전통적 for는 초기화·조건·증감식을 한 문장에 두고, Python의 for는 iterable에서 원소를 하나씩 꺼내 변수에 바인딩한다. `range(start, stop, step)`가 정수 수열을 제공해 C식 카운터 반복과 비슷한 모양을 만든다.

exit condition과 무한 루프

반복을 끝내는 조건이 절대로 참이 되지 않거나 상태 갱신이 빠지면 무한 루프가 된다. 서버의 event loop처럼 의도적 무한 루프도 있으며, 이 경우 signal, socket close, shutdown flag 같은 별도 종료 경로를 설계한다.

pass·continue·break

pass

문법적으로 문장이 필요하지만 아무 동작도 하지 않는 placeholder다.

continue

현재 반복의 남은 본문을 건너뛰고 다음 반복으로 간다.

break

가장 가까운 반복문을 즉시 종료한다.

unreachable code

항상 return, raise, break 뒤에 있어 실행될 수 없는 코드다. 논리 오류이거나 리팩터링 잔재일 수 있다. 보안 코드에서는 검증이 return 뒤에 놓여 실제로 실행되지 않는 실수가 치명적일 수 있다.

반복 흐름 제어

```
for i in range(10):
    if i % 2 == 0:
        continue
    if i > 7:
        break
    print(i)

while True:
    event = receive_event()
    if event is None:      # 명시적 종료 조건
        break
    handle(event)
```

파일 입출력·Context Manager·Race Condition

필기 확인 + 보충 해설

파일을 "문자열 변수"가 아니라 유한한 OS 자원으로 보기

함께 찾을 말: open, read, write, append, close, with, context manager, __enter__, __exit__, race condition, cursor

파일 입출력·Context Manager·Race Condition란 파일 입출력은 커널의 열린 파일 객체와 file descriptor를 통해 바이트를 읽고 쓰는 작업이고, context manager는 자원 획득과 해제를 예외 안전하게 묶으며, race condition은 실행 순서에 따라 공유 상태 결과가 달라지는 문제다.

파일 open은 커널 객체를 얻는 과정이다

Python의 open()은 경로를 해석하고 권한을 검사해 커널의 열린 파일 상태를 만들고 file descriptor를 받는다. Python file object는 그 descriptor와 인코딩·버퍼링을 감싼다. 모드 r, w, a, x와 +는 읽기·쓰기·생성·덮어쓰기·append 정책을 정한다.

왜 close가 필요한가

열린 파일·socket은 커널 테이블 항목, 버퍼, 포트, 락 같은 유한 자원을 사용한다. 출력은 사용자 공간 buffer에 남아 있을 수 있어 close 시 flush된다. close하지 않으면 descriptor 고갈, 데이터 유실, 연결 미종료가 생길 수 있다. garbage collection 시점에 맡기지 말고 context manager로 생명주기를 명시하는 편이 안전하다.

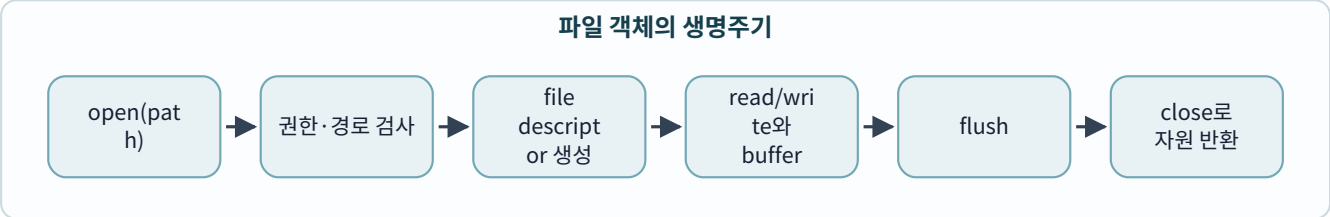


그림 4-4. with 블록은 정상 종료뿐 아니라 예외 발생 시에도 __exit__을 통해 정리한다.

Context manager와 __enter__/__exit__

`with`는 자원 획득과 해제를 짝으로 보장하는 프로토콜이다. 객체의 `__enter__`가 사용할 자원을 반환하고, 블록 종료 시 `__exit__`가 호출된다. 양옆에 double underscore가 붙은 이름은 Python 데이터 모델이 특별한 문법과 연산에 연결하는 special method다.

context manager로 자원 닫기

```
from pathlib import Path

path = Path("events.log")
with path.open("a", encoding="utf-8") as fp:
    fp.write("login success\n")
# 여기서 예외 여부와 상관없이 파일 close

# 소켓도 같은 원리
with socket.create_connection((host, port), timeout=3) as sock:
    sock.sendall(payload)
```

Race condition

여러 실행 흐름이 공유 상태를 읽고 수정하는 순서에 따라 결과가 달라지는 현상이다. “파일이 없으면 만든다”를 check 후 create 두 단계로 나누면 사이에 다른 프로세스가 만들 수 있다. 원자적 생성 모드, file lock, transaction, DB unique constraint처럼 경쟁을 고려한 기능을 사용해야 한다.

실행 인자와 Python 가상 환경

필기 확인 + 보충 해설

재현 가능한 실행 입력과 의존성 구성

함께 찾을 말: argv, argparse, venv, virtual environment, Anaconda, Jupyter Notebook

실행 인자와 Python 가상 환경란 실행 인자는 프로세스 시작 시 전달되는 문자열 값이고, Python 가상 환경은 프로젝트마다 interpreter와 패키지 설치 경로를 분리하는 경량 격리 구조다.

argv는 프로세스 시작 시 전달된 문자열 목록이다

셸에서 `python scan.py 10.0.0.1 --port 443`를 실행하면 OS가 프로그램에 argument vector를 전달하고 Python은 이를 `sys.argv`로 제공한다. 모두 문자열이므로 숫자 검증과 변환이 필요하다. argparse를 쓰면 옵션, 도움말, 필수값을 구조화할 수 있다.

가상 환경

Python virtual environment는 프로젝트별 interpreter 경로와 package 설치 위치를 분리해 서로 다른 버전의 의존성이 충돌하지 않게 한다. VM처럼 별도 kernel을 만드는 것이 아니며 OS의 Python 실행 환경을 디렉터리 수준에서 격리한다.

Python 가상 환경

```
python3 -m venv .venv
source .venv/bin/activate
python -m pip install --upgrade pip
python -m pip install requests
python -m pip freeze > requirements.txt

deactivate
```

Anaconda와 Jupyter

Anaconda/conda는 Python뿐 아니라 native library까지 환경별로 관리하는 배포·패키지 생태계다. Jupyter Notebook은 코드·설명·출력을 셀 단위로 실행하는 대화형 문서 환경이며, production 서비스 실행 방식과는 구분해야 한다.

5장. 가상화·VMware·GNS3: 실습 환경은 무엇을 흉내 내는가

> 가상 머신과 네트워크 실습 도구는 모두 “가짜 환경”이지만 같은 방식은 아니다. 시뮬레이션은 동작 모델을 계산하고, 에뮬레이션은 실제 장비용 소프트웨어가 실행될 수 있는 환경을 재현하며, 가상화는 실제 CPU·메모리·I/O 자원을 격리해 여러 시스템에 나눈다.

가상화·Hypervisor·VM

필기 확인 + 보충 해설

한 대의 컴퓨터에서 여러 운영체제가 독립적으로 보이는 이유

함께 찾을 말: Virtualization, Hypervisor, Full Virtualization, Paravirtualization, Hypercall, Bare metal, VMware ESXi, Xen

가상화·Hypervisor·VM란 가상화는 하나의 물리 자원을 여러 격리된 논리 자원처럼 제공하는 기술이고, hypervisor는 VM의 CPU·메모리·디스크·NIC를 물리 하드웨어에 매핑하고 통제하는 계층이다.

Hypervisor가 중개하는 자원

hypervisor는 VM마다 가상 CPU, 메모리, 디스크, NIC를 제공하고 실제 하드웨어 자원에 매핑한다. CPU의 가상화 확장과 page table 보조, I/O device emulation 또는 paravirtualized driver를 활용한다. VM 안의 guest OS는 자신이 독립 컴퓨터라고 인식한다.

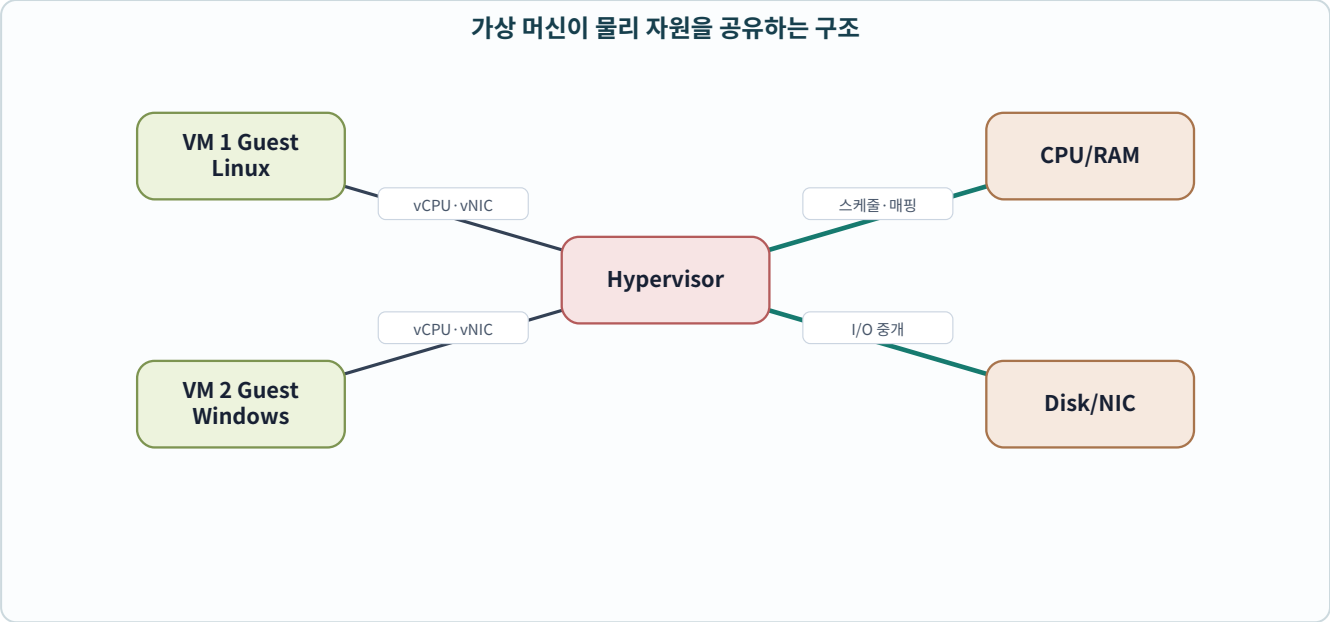


그림 5-1. VM 격리는 강력하지만 hypervisor, 관리 plane, 공유 하드웨어는 공통 신뢰 경계다.

전가상화와 반가상화

전가상화(full virtualization)는 guest가 일반 하드웨어에서 동작하는 것처럼 보이게 하며 수정 없는 OS 실행을 목표로 한다. 반가상화(paravirtualization)는 guest가 hypervisor와 협력하는 hypercall·가상화 전용 driver를 사용해 오버헤드를 줄인다. 현대 시스템은 CPU는 hardware-assisted virtualization, I/O는 virtio 같은 paravirtual driver를 함께 쓰는 혼합 형태가 흔하다.

Type 1·Type 2·Bare metal

Type 1 hypervisor는 하드웨어 위에 직접 설치되는 형태로 VMware ESXi, Xen 계열이 대표적이고, Type 2는 일반 host OS 위의 애플리케이션처럼 실행되는 VMware Workstation, VirtualBox가 대표적이다. bare metal은 중간 범용 OS 계층 없이 물리 하드웨어에 직접 배치한다는 뜻으로 서버·hypervisor 문맥에 따라 사용된다.

ISO 이미지와 VM Snapshot

필기 확인 + 보충 해설

설치 매체와 되돌리기 지점을 backup과 구분하기

함께 찾을 말: ISO image, snapshot, clone

ISO 이미지와 VM Snapshot란 ISO 이미지는 광학 디스크 구조를 파일로 보존한 이미지이고, snapshot은 VM의 디스크·메모리 상태를 특정 시점 기준으로 기록해 되돌릴 수 있게 하는 기능이다.

ISO 이미지 파일

ISO image는 광학 디스크의 파일시스템과 내용을 하나의 파일로 보존한 디스크 이미지다. VM의 가상 CD/DVD 장치에 연결하면 guest는 설치 디스크가 삽입된 것처럼 부팅할 수 있다. 단순 압축파일이 아니라 부팅 구조와 파일시스템 메타데이터를 포함할 수 있다.

Snapshot

snapshot은 특정 시점의 VM 디스크 상태와 선택적으로 메모리·장치 상태를 기준점으로 저장해 되돌릴 수 있게 한다. 일반적으로 base disk에 변경분(delta)을 이어 쓰는 구조이므로 장기간 쌓으면 성능·용량·체인 손상 위험이 커진다. snapshot은 별도 매체·별도 장애영역에 보존하는 backup을 대체하지 않는다.



그림 5-2. 원본과 snapshot이 같은 datastore에 있으면 저장장치 장애 때 함께 잃을 수 있다.

복제와 snapshot

clone은 독립 또는 연결된 새 VM 인스턴스를 만드는 기능이고 snapshot은 같은 VM의 시점 상태를 관리하는 기능이다. 제품별 구현과 용어가 다르므로 운영 전 공식 문서를 확인한다.

VM 네트워크의 NAT·Bridge·Host-only

필기 확인 + 보충 해설

가상 NIC가 어느 브로드캐스트 도메인에 연결되는가

함께 찾을 말: NAT mode, Bridged mode, Host-only, virtual switch, vNIC

VM 네트워크의 NAT·Bridge·Host-only란 VM NAT는 host가 주소를 변환해 guest의 외부 통신을 중개하고, bridge는 guest vNIC를 물리 LAN의 L2 구성원처럼 연결하며, host-only는 외부와 분리된 host-VM 네트워크를 만든다.

NAT 모드

guest가 사설 가상 네트워크 주소를 쓰고 host의 NAT를 통해 외부로 나간다. 외부에서는 기본적으로 guest에 직접 새 연결을 열기 어렵고 port forwarding이 필요하다. 실습 VM의 인터넷 접근은 쉽지만 실제 LAN의 독립 장비처럼 보이지 않는다.

Bridge 모드

가상 NIC를 host의 물리 L2 네트워크에 bridge해 guest가 같은 LAN의 독립 장치처럼 DHCP를 받거나 IP를 설정한다. 무선 NIC·기업 NAC·promiscuous 제한 때문에 제품과 환경별 제약이 있을 수 있다.

Host-only와 내부 네트워크

host-only는 host와 VM 사이의 격리된 가상 LAN을 만들고, internal/private network는 VM들끼리만 통신하도록 구성할 수 있다. 공격 실습은 외부망으로 새지 않도록 격리망과 snapshot을 사용해야 한다.

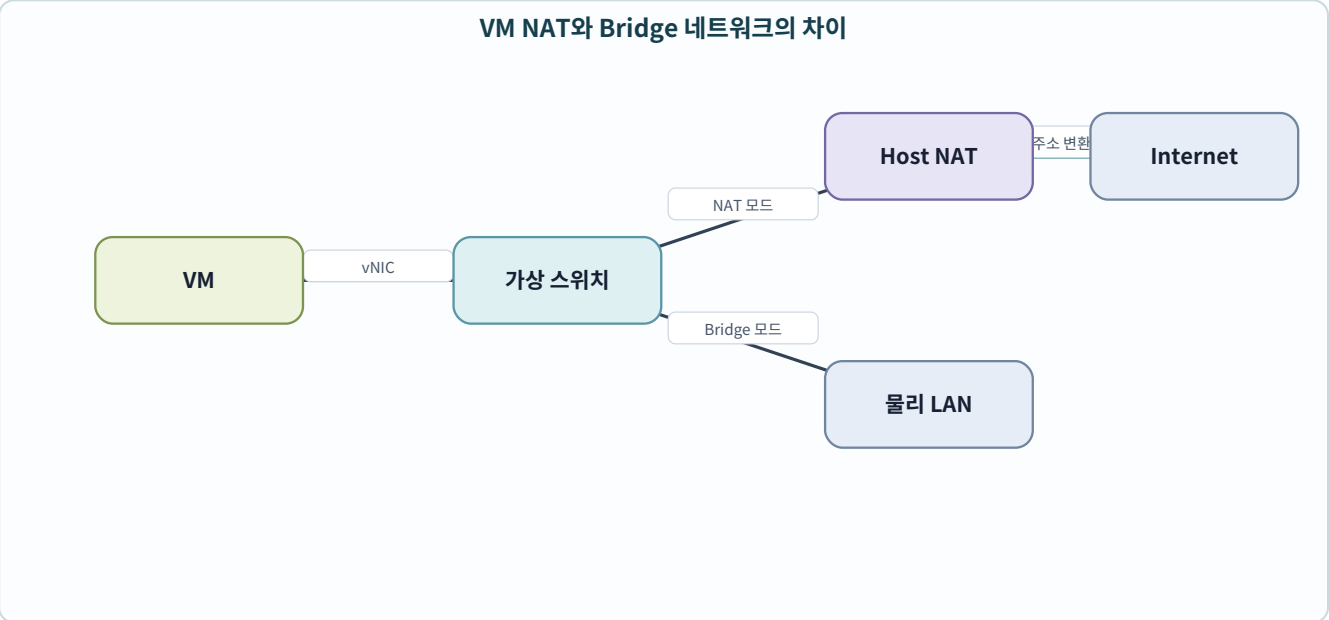


그림 5-3. NAT와 bridge는 “인터넷 됨/안 됨”보다 L2 경계와 주소가 어디에 존재하는지로 구분한다.

GNS3·에뮬레이션·Cisco IOS CLI

필기 확인 + 보충 해설

실습 화면 뒤에서 실제로 어떤 프로세스와 이미지가 동작하는가

함께 찾을 말: GNS3, VPCS, QEMU, IOSv, IOU, Cisco IOS, enable, configure terminal, no shutdown, running-config, startup-config

GNS3·에뮬레이션·Cisco IOS CLI란 GNS3는 여러 가상화·에뮬레이션 backend와 가상 링크를 조합해 네트워크 topology를 실행하는 플랫폼이며, Cisco IOS CLI는 권한·설정 범위를 mode로 나누어 장비 상태를 조회하고 변경한다.

Simulation과 Emulation

simulation은 실제 시스템의 동작을 수학적·논리적 모델로 계산하므로 빠르고 대규모 실험에 유리하지만 실제 OS 이미지의 모든 버그와 CLI를 재현하지 않을 수 있다. emulation은 목표 CPU·장치 환경을 흉내 내 실제 장비용 binary/OS를 실행하므로 현실성이 높지만 자원 요구와 라이선스 제약이 크다.

GNS3의 구성 요소

GNS3 GUI는 topology를 설계하고, local/remote GNS3 server가 QEMU, Dynamips, Docker, VPCS 같은 backend를 실행한다. VPCS는 가벼운 가상 PC, QEMU는 다양한 VM 이미지, IOS on Unix/IOSv 계열은 Cisco 실습 이미지에 사용된다. Cloud node는 host interface나 VMnet을 topology에 연결한다.

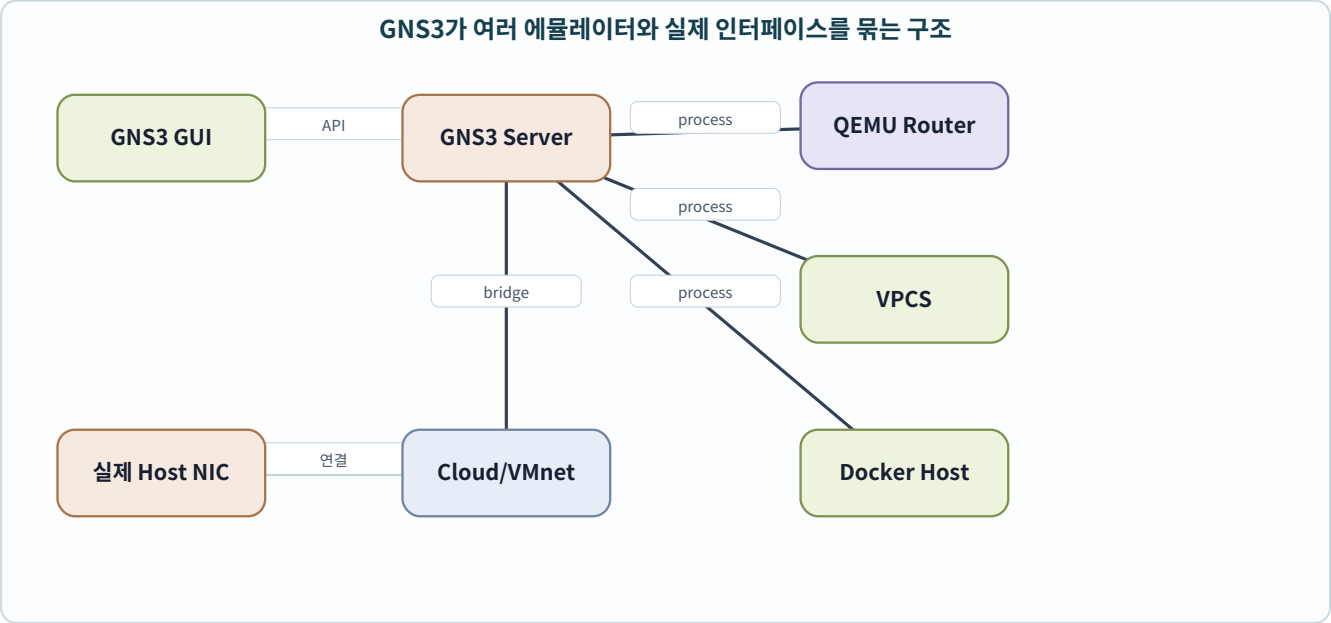


그림 5-4. GNS3 자체가 모든 라우터를 구현하는 것이 아니라 여러 backend를 orchestration한다.

인터페이스 이름

Linux의 `eth0`, `ens160`, `enp0s3`는 NIC를 가리키는 논리 이름이다. predictable network interface name은 firmware/PCI 위치를 반영해 장치 순서 변화에도 이름이 안정되도록 한다. Cisco의 `GigabitEthernet0/1`도 장비 내 인터페이스 식별자다. 인터페이스는 물리 카드 전체, 포트, 가상 subinterface를 문맥에 따라 가리킨다.

Cisco CLI mode

user EXEC(>)는 제한된 조회, privileged EXEC(#)는 폭넓은 조회·관리, global configuration은 장비 전역 설정, interface/line/router submode는 특정 구성 요소 설정을 담당한다. no는 많은 명령에서 해당 설정을 제거하거나 반대 상태로 되돌리는 문법이다. `shutdown`은 관리적으로 인터페이스를 내려 데이터 전달을 막고, `no shutdown`은 관리적으로 활성화한다.

Cisco IOS 기본 모드 전환

```
enable
configure terminal
hostname SW1
interface gigabitEthernet 0/1
 description USER-PC
 no shutdown
end
show ip interface brief
copy running-config startup-config
```

running-config와 startup-config

running-config는 현재 RAM에서 적용 중인 설정, startup-config는 재부팅 시 읽을 비휘발 저장 설정이다. `write memory` 또는 `copy running-config startup-config`는 현재 설정을 부팅 설정으로 저장한다. 저장하지 않으면 재부팅 후 사라질 수 있다.

On-premises·Cloud·IaC·Container

필기 확인 + 보충 해설

가상화 이후 인프라를 배치하고 반복 관리하는 방법

함께 찾을 말: On-premises, Cloud, Infrastructure as Code, Terraform, Ansible, Docker, Container, OpenStack, resource pool, VDI

On-premises·Cloud·IaC·Container란 on-premises와 cloud는 인프라 소유·제공·책임 모델을 구분하고, IaC는 구성을 코드로 재현하며, container는 host kernel을 공유하는 process 격리 방식이다.

On-premises와 Cloud

on-premises는 조직이 통제하는 자체 시설이나 전용 환경에 인프라를 배치·운영하는 모델이고, cloud는 공급자가 자원 pool과 API를 제공해 필요할 때 할당하는 모델이다. 둘의 차이는 단순 위치가 아니라 책임 분담, 과금, 확장, 통제 방식이다.

Infrastructure as Code

IaC는 서버·네트워크·권한·cloud 자원 구성을 사람이 GUI로 한 번씩 클릭하는 대신 선언 파일과 코드로 관리해 version control, review, 반복 배포, drift 탐지를 가능하게 한다. Terraform은 주로 자원 상태를 선언·프로비저닝하고, Ansible은 host 설정·패키지·서비스를 원하는 상태로 만드는 데 흔히 사용된다. 경계는 절대적이지 않다.

Container와 VM

container는 host kernel을 공유하면서 process, filesystem, network namespace와 cgroup으로 격리한다. VM은 별도 guest kernel을 가진다. container가 가볍지만 kernel 취약점과 host 설정이 공통 경계다. Docker는 image, container runtime, network, volume을 관리하는 생태계다.

OpenStack과 resource pool

OpenStack은 compute, network, block/object storage, identity 같은 데이터센터 자원을 API로 묶어 private cloud를 구성하는 오픈소스 플랫폼이다. resource pool은 여러 물리 자원을 논리적으로 모아 수요에 따라 할당하는 개념이다.

제2부

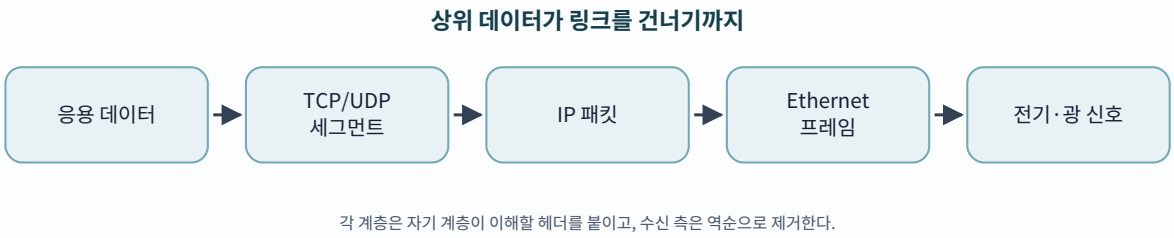
같은 LAN 안에서 프레임이 움직이는 법

이 부의 중심 질문은 하나다. “IP 패킷을 만들기 전에, 한 장치가 같은 네트워크의 다른 장치까지 실제 전기·광 신호와 Ethernet 프레임을 어떻게 전달하는가?” NIC, MAC 주소, 스위치, 브로드캐스트, VLAN, 802.1Q, STP, 포트 시큐리티를 이 질문에 차례대로 연결한다. 용어를 외우기보다 프레임 한 개가 스위치 포트에 들어온 순간부터 다음 포트로 나갈 때까지를 추적한다.

6장. Ethernet 프레임이 만들어지고 스위치까지 도착하는 과정

컴퓨터끼리 통신한다고 말하면 흔히 IP 주소부터 떠올리지만, 유선 LAN의 실제 링크 위에는 Ethernet 프레임이 흐른다. IP 패킷은 혼자 케이블을 건너지 못한다. NIC가 IP 패킷을 Ethernet 프레임의 데이터 영역에 넣고, 목적지·출발지 MAC 주소와 오류 검사용 정보를 붙인 뒤 신호로 바꿔 내보낸다. 이 장은 그 포장 과정을 아주 낮은 계층부터 설명한다.

이 장의 연결 지도



Ethernet

필기 확인 + IEEE/RFC 보강

LAN을 구현하는 대표 기술이지 LAN 그 자체는 아니다

함께 찾을 말: 이더넷, IEEE 802.3

Ethernet란 유선 LAN에서 장치가 프레임 형식, MAC 주소, 매체 접근, 신호 전송 규칙을 공통으로 사용하도록 정한 기술군이다.

말뜻부터 이해하기

Ethernet은 특정 회사 제품명이 아니라 IEEE 802 계열에서 표준화된 유선 LAN 기술군을 가리킨다. “이더넷 = LAN”은 아니다. LAN은 가까운 범위의 네트워크라는 범주이고, Ethernet은 그 LAN을 구현하는 대표적인 데이터링크·물리 계층 기술이다. 같은 LAN을 Wi-Fi로 만들 수도 있으며, 오래된 Token Ring처럼 다른 기술로 만들 수도 있었다.

왜 필요한가

케이블에 단순히 비트를 흘려보내기만 하면 수신자는 “어디부터 어디까지가 한 메시지인지”, “누가 보냈는지”, “누가 받아야 하는지”, “전송 중 비트가 망가졌는지”를 알 수 없다. Ethernet은 한 번에 전달할 데이터 덩어리를 **프레임(frame)**으로 구획하고, 주소와 타입, 오류 검출 정보를 일정한 형식으로 붙여 서로 다른 제조사의 장비가 같은 규칙으로 통신하게 한다.

Ethernet II 프레임의 큰 구조

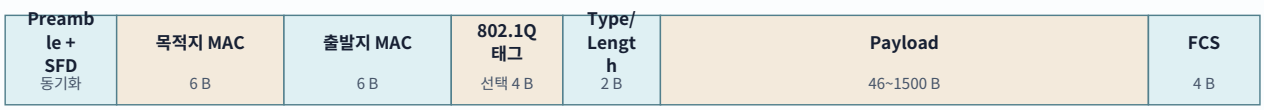


그림 6-1. 프레임은 주소, 상위 프로토콜 식별자, 실제 데이터, 오류 검출값으로 구성된다. 802.1Q 태그는 VLAN 트렁크 등에서 선택적으로 삽입된다.

프레임이 만들어지는 순서

1. **응용 프로그램**이 보낼 데이터를 소켓에 쓴다.
2. 운영체제의 TCP 또는 UDP가 포트 번호 등의 전송 계층 정보를 붙인다.
3. IP 계층이 출발지·목적지 IP 주소와 TTL 같은 정보를 붙여 IP 패킷을 만든다.
4. 라우팅 판단 결과에 따라 “다음 홉”을 정하고, ARP 또는 이웃 탐색으로 그 다음 홉의 링크 계층 주소를 알아낸다.
5. NIC 드라이버와 NIC가 목적지 MAC, 출발지 MAC, EtherType, FCS 등을 포함한 Ethernet 프레임을 구성한다.
6. NIC의 PHY가 비트를 전기 신호 또는 광 신호로 바꾸어 링크에 내보낸다.

중요한 구분: 패킷과 프레임

패킷(packet)은 넓은 뜻으로 네트워크 데이터 단위를 부르는 말이지만, 계층을 정확히 말할 때 IPv4/IPv6의 데이터 단위를 IP 패킷, Ethernet의 데이터 단위를 프레임이라고 한다. IP 패킷은 라우터를 지날 때 대체로 유지되지만, 각 링크 구간의 Ethernet 헤더는 매 홉 새로 만들어진다.

FCS와 CRC가 하는 일

FCS(Frame Check Sequence)는 프레임 뒤에 붙는 오류 검출 값이다. 송신 측은 프레임 비트열에 CRC 계산을 적용해 값을 만들고, 수신 측은 같은 계산을 다시 수행한다. 값이 맞지 않으면 전송 중 비트 오류가 있었다고 보고 프레임을 버린다. 이것은 "잘못된 비트를 자동으로 고치는 기능"이 아니다. Ethernet FCS는 주로 오류를 **검출**하며, 재전송 여부는 상위 계층(TCP 등)이나 응용이 결정한다.

캡처에서 FCS가 안 보일 수 있는 이유

많은 NIC는 FCS를 하드웨어에서 검사한 뒤 운영체제에 넘길 때 제거한다. 따라서 Wireshark에서 FCS 필드가 보이지 않는다고 "Ethernet에 FCS가 없다"고 판단하면 안 된다. 캡처 위치와 NIC/드라이버 설정을 확인해야 한다.

MAC 주소

필기 확인 + IEEE/IANA 보강

IP 주소와 역할·범위가 다르다

함께 찾을 말: Media Access Control Address, 물리 주소, OUI, UAA

MAC 주소란 같은 링크 또는 VLAN 안에서 Ethernet 프레임의 송신·수신 인터페이스를 식별하는 보통 48비트의 링크 계층 주소다.

MAC이라는 단어의 의미

MAC은 **Media Access Control**, 즉 "공유 매체에 누가 어떤 방식으로 접근할지 제어하는 것"을 뜻한다. Ethernet에서 MAC 주소는 프레임을 같은 링크 안의 어느 인터페이스로 보낼지 식별하는 링크 계층 주소다.

48비트 주소의 구조

일반적인 Ethernet MAC 주소는 48비트(6바이트)이며, 사람이 읽기 좋게 `00:11:22:33:44:55`처럼 16진수로 표시한다. 앞부분은 제조사에 할당된 OUI와 연결될 수 있고, 뒤쪽은 제조사가 인터페이스를 구분하도록 정한다. 다만 현대 운영체제의 임의 MAC, 가상 NIC, 클라우드 환경에서는 "앞 24비트만 보면 항상 실제 제조사를 알 수 있다"는 가정이 깨질 수 있다.

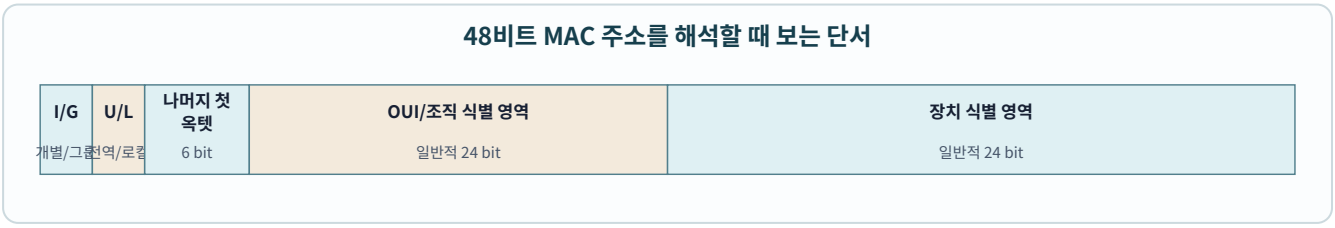


그림 6-2. 최하위 비트의 I/G와 U/L 의미, OUI 영역, 장치 식별 영역을 구분한다. 실제 구현에서는 로컬 관리 주소와 무작위 주소가 흔하다.

유니캐스트·멀티캐스트·브로드캐스트

유니캐스트 MAC

한 인터페이스를 대상으로 하는 개별 주소다.

멀티캐스트 MAC

특정 그룹을 대상으로 하며, 첫 옥텟의 I/G 비트가 그룹 주소임을 나타낸다.

브로드캐스트 MAC

모든 비트가 1인 `ff:ff:ff:ff:ff:ff`이며 같은 VLAN의 모든 포트로 확산된다.

> **MAC 주소는 인터넷 전체의 경로 주소가 아니다**
>
> MAC 주소는 현재 링크에서 프레임을 전달하는 데 사용한다. 라우터를 지나 다음 Ethernet 링크로 넘어가면 출발지·목적지 MAC 주소는 그 링크에 맞게 다시 작성된다. 인터넷 전체에서 목적지 네트워크를 찾는 역할은 IP 주소와 라우팅이 맡는다.

실제로 확인하기

MAC 주소와 이웃 정보를 보는 명령

```
# Linux 인터페이스와 MAC 주소 보기
ip -br link
ip link show dev eth0

# 이웃 IP와 MAC 매핑 보기
ip neigh show

# Wireshark 표시 필터 예시
eth.addr == 00:11:22:33:44:55
```

NIC

필기 확인 + Linux 보강

소프트웨어 패킷을 실제 신호로 바꾸는 경계

함께 찾을 말: 랜카드, Network Interface Controller, 네트워크 인터페이스

NIC란 컴퓨터의 네트워크 스택과 전기·광 링크 사이에서 프레임 송수신, 큐잉, DMA, 링크 협상 등을 수행하는 네트워크 인터페이스 장치다.

NIC가 하는 일

NIC(Network Interface Controller/Card)는 운영체제의 네트워크 스택과 물리 링크 사이를 연결한다. 소프트웨어가 만든 패킷을 DMA를 통해 송신 큐로 받고, Ethernet 헤더·체크섬·분할 작업 일부를 하드웨어로 처리할 수 있으며, PHY를 통해 전기·광 신호를 송수신한다. 수신 시에는 반대로 신호를 비트열로 복원하고 프레임을 검사해 메모리의 RX 링에 놓는다.

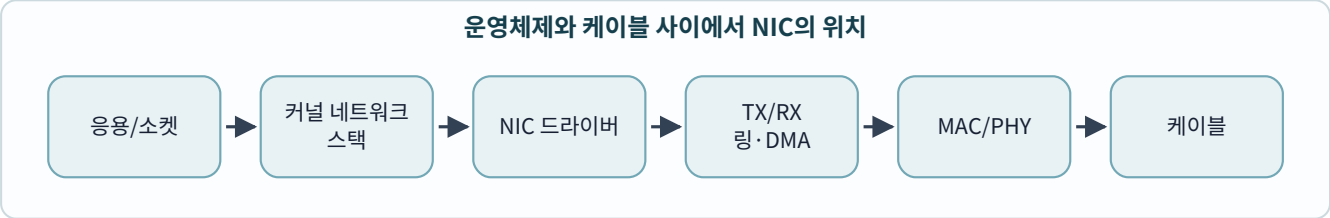


그림 6-3. NIC는 단순한 "랜선 꽂는 구멍"이 아니라 큐, DMA, 오프로딩, 링크 협상 기능을 가진 장치다.

인터페이스와 NIC는 같은 말인가

물리 장비에서는 인터페이스가 NIC의 포트와 거의 겹쳐 보이지만, 운영체제의 인터페이스는 더 넓은 개념이다. `eth0`, `ens160` 같은 물리/가상 NIC 인터페이스 외에도 `lo` 루프백, VLAN 서브인터페이스, 브리지, 터널, 본딩 인터페이스가 존재한다. 즉, "인터페이스 = 랜카드"는 입문 단계의 근사이며 실무에서는 논리 인터페이스까지 포함한다.

링크 상태란

링크 상태(link state)는 물리적으로 신호를 주고받을 수 있고 속도·듀플렉스 등의 링크 협상이 성립했는지를 뜻한다. 케이블이 빠졌거나 상대 포트가 shutdown이면 물리 계층이 down일 수 있다. 반대로 링크가 up이어도 VLAN, IP, 라우팅, 방화벽 문제 때문에 통신은 실패할 수 있다.

Linux에서 링크·속도·드라이버 통계를 확인

```
ip -br link
ethtool eth0
ethtool -S eth0 | less
cat /sys/class/net/eth0/operstate
```

캡슐화와 역캡슐화

필기 확인 + 프로토콜 보강

계층마다 문제를 나누어 해결하는 포장 규칙

함께 찾을 말: Encapsulation, Decapsulation, PDU

캡슐화와 역캡슐화란 상위 계층 데이터를 하위 계층의 데이터 영역에 넣고 제어 헤더를 붙이는 과정과, 수신 측에서 이를 역순으로 검사·제거하는 과정이다.

캡슐화가 필요한 이유

각 계층은 자신이 해결해야 하는 문제만 다룬다. HTTP는 문서 요청 형식을, TCP는 신뢰성 있는 바이트 흐름을, IP는 네트워크 사이의 경로를, Ethernet은 한 링크의 프레임 전달을 맡는다. 상위 계층 데이터 앞에 해당 계층의 제어 정보를 붙이는 과정을 캡슐화라고 한다.

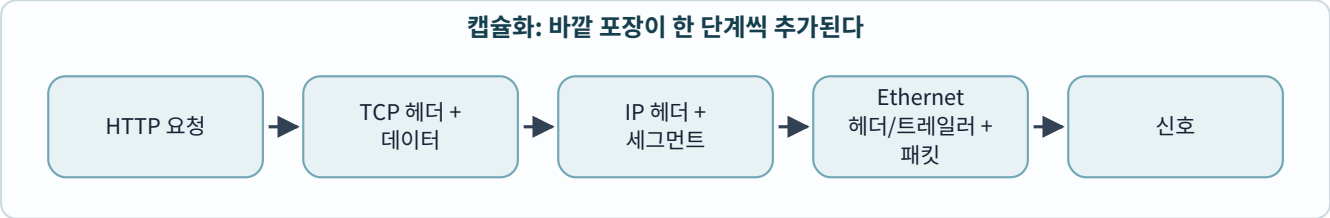


그림 6-4. 수신 측에서는 바깥쪽 프레임부터 검사하고 헤더를 제거해 위 계층으로 올린다.

라우터에서 무엇이 바뀌는가

라우터는 들어온 Ethernet 프레임의 FCS와 목적지 MAC을 확인한 뒤 L2 헤더를 제거하고 IP 패킷을 처리한다. 목적지 IP에 맞는 다음 홉을 선택하고 TTL을 감소시키며 헤더 체크섬을 갱신한 뒤, 나가는 인터페이스에 맞는 새 L2 프레임을 만든다. 따라서 한 종단 간 통신에서도 링크마다 MAC 주소와 FCS는 달라진다.

“캡슐화 해제 후 원본이 그대로 나온다”의 범위

수신 종단에서는 상위 데이터가 복원되지만, 중간 라우터에서는 IP TTL, 체크섬, NAT가 적용되면 주소·포트 등이 바뀔 수 있다. 계층 모델은 이해를 위한 경계이며 실제 장비는 오프로딩, 터널, 방화벽 같은 기능으로 여러 계층을 함께 처리한다.

7장. 스위치는 프레임을 어떻게 배우고 전달하는가

> 스위치는 목적지 MAC 주소만 보고 즉시 정답 포트를 아는 장비가 아니다. 먼저 들어온 프레임의 출발지 MAC 주소를 보고 “이 주소는 이 포트 뒤에 있다”고 학습한다. 목적지 위치를 알고 있으면 한 포트로 전달하고, 모르면 같은 VLAN의 여러 포트로 복제한다. 이 단순한 규칙이 정상 전달, flooding, MAC table aging, MAC flapping, L2 loop를 모두 설명한다.

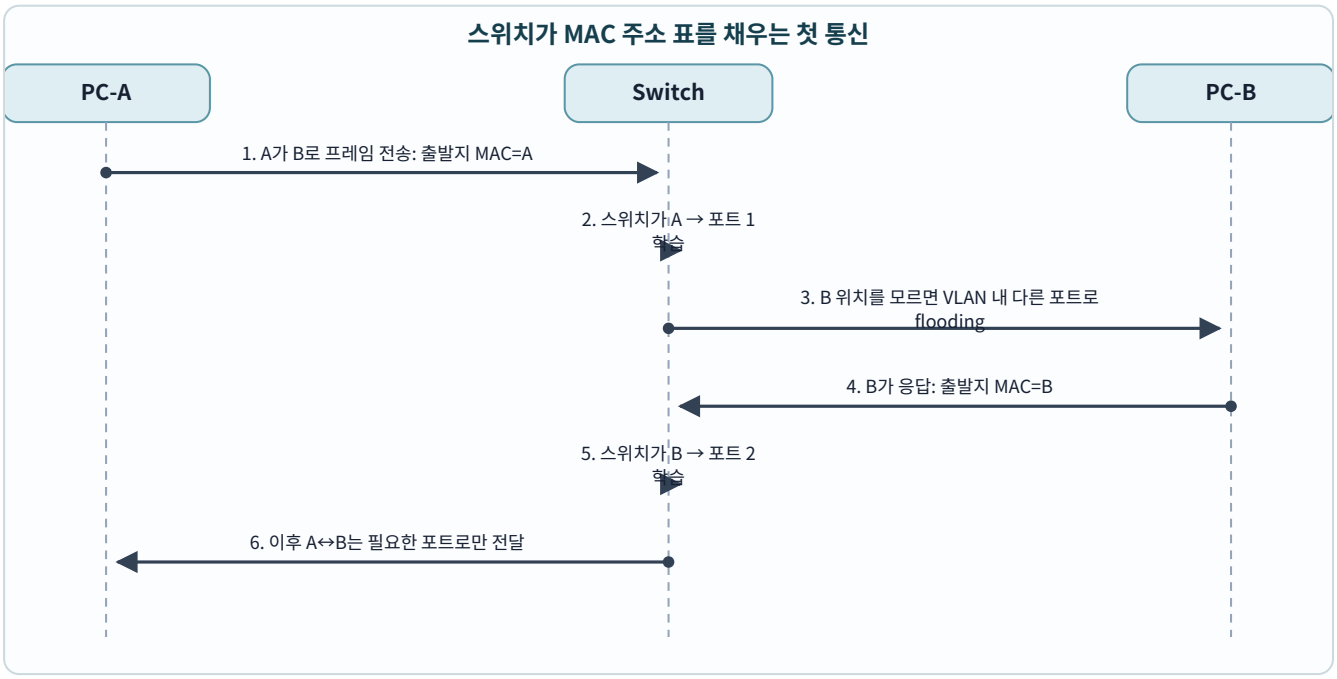


그림 7-1. 스위치는 수신 프레임의 출발지 주소를 학습한다. 목적지 주소는 전달 결정을 위해 조회한다.

스위치와 투명 브리징

필기 확인 + IEEE/Cisco 보강

호스트에게 “투명하게” MAC 기반 전달을 수행한다

함께 찾을 말: Switch, Bridge, Transparent Bridging, 브리지

스위치와 투명 브리징란 호스트가 중간 경로를 따로 설정하지 않아도, 수신 프레임의 출발지 MAC을 학습하고 목적지 MAC에 따라 같은 VLAN의 적절한 포트로 프레임을 전달하는 L2 기능이다.

브리지와 스위치의 관계

브리지(bridge)는 여러 L2 세그먼트를 연결하고 MAC 주소를 학습해 프레임을 선택적으로 전달하는 장치 또는 기능이다. 현대의 Ethernet 스위치는 다수 포트와 하드웨어 처리 능력을 가진 멀티포트 브리지로 이해할 수 있다. Linux bridge도 같은 원리를 소프트웨어로 구현한다.

스위치의 핵심 세 단계

1. **학습(learning)**: 프레임이 들어온 VLAN과 포트, 출발지 MAC을 표에 기록한다.
2. **조회(lookup)**: 목적지 MAC이 같은 VLAN의 표에 있는지 찾는다.
3. **전달/필터링**: 목적지가 다른 포트면 그 포트로 전달하고, 같은 수신 포트 뒤라면 다시 보내지 않으며, 모르면 flooding한다.

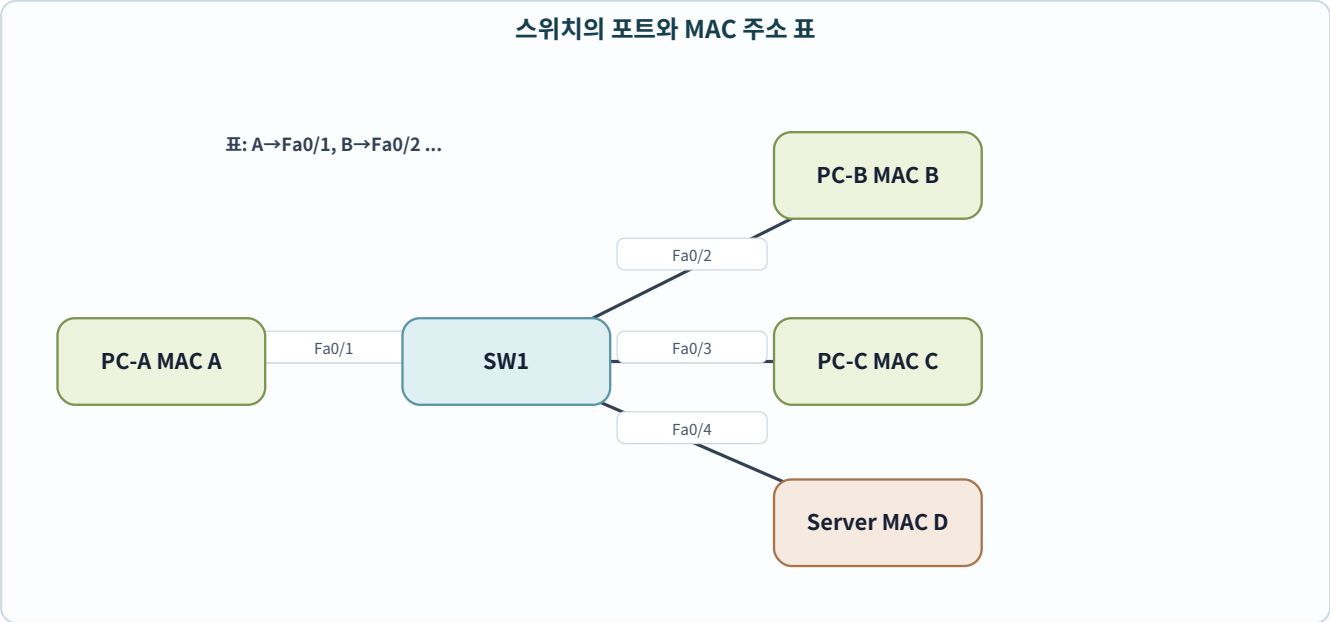


그림 7-2. MAC 주소 표는 보통 VLAN + MAC 주소를 포트에 연결한다. ARP 표와 다른 표다.

MAC 주소 표와 ARP 표를 섞지 않기

스위치 MAC 주소 표는 “VLAN 안의 MAC 주소가 어느 스위치 포트 뒤에 있는가”를 저장한다. **호스트·라우터의 ARP/neighbor 표**는 “이웃 IP 주소가 어느 MAC 주소인가”를 저장한다. 하나는 MAC→포트, 다른 하나는 IP→MAC이다.

MAC 학습 결과 확인

```
# Cisco 계열 예시
show mac address-table
show mac address-table dynamic
show mac address-table interface gigabitEthernet 0/1

# Linux bridge 예시
bridge fdb show
bridge link show
```

Flooding과 MAC 주소 표 에이징

필기 확인 + 스위칭 보강

정상 동작이지만 루프가 만나면 폭주한다

함께 찾을 말: 플러딩, Unknown Unicast Flooding, MAC Table Aging

Flooding과 MAC 주소 표 에이징란 Flooding은 목적지 위치를 모르거나 전체 전달이 필요한 프레임을 같은 VLAN의 여러 포트로 복제하는 동작이고, aging은 오래 사용되지 않은 동적 MAC 위치 정보를 지우는 동작이다.

Flooding이라는 말뜻

Flooding은 “물이 넘쳐 퍼진다”는 뜻처럼, 스위치가 프레임을 특정 한 포트로 결정하지 못해 같은 VLAN의 여러 포트로 복제해 보내는 동작이다. 수신 포트로는 다시 내보내지 않는다.

언제 flooding하는가

브로드캐스트

목적지 MAC이 ff:ff:ff:ff:ff:ff이므로 VLAN 전체에 전달한다.

Unknown unicast

유니캐스트 목적지 MAC이지만 표에 위치가 없어서 찾기 위해 확산한다.

일부 멀티캐스트

IGMP snooping 등 최적화 정보가 없으면 VLAN 여러 포트로 확산될 수 있다.

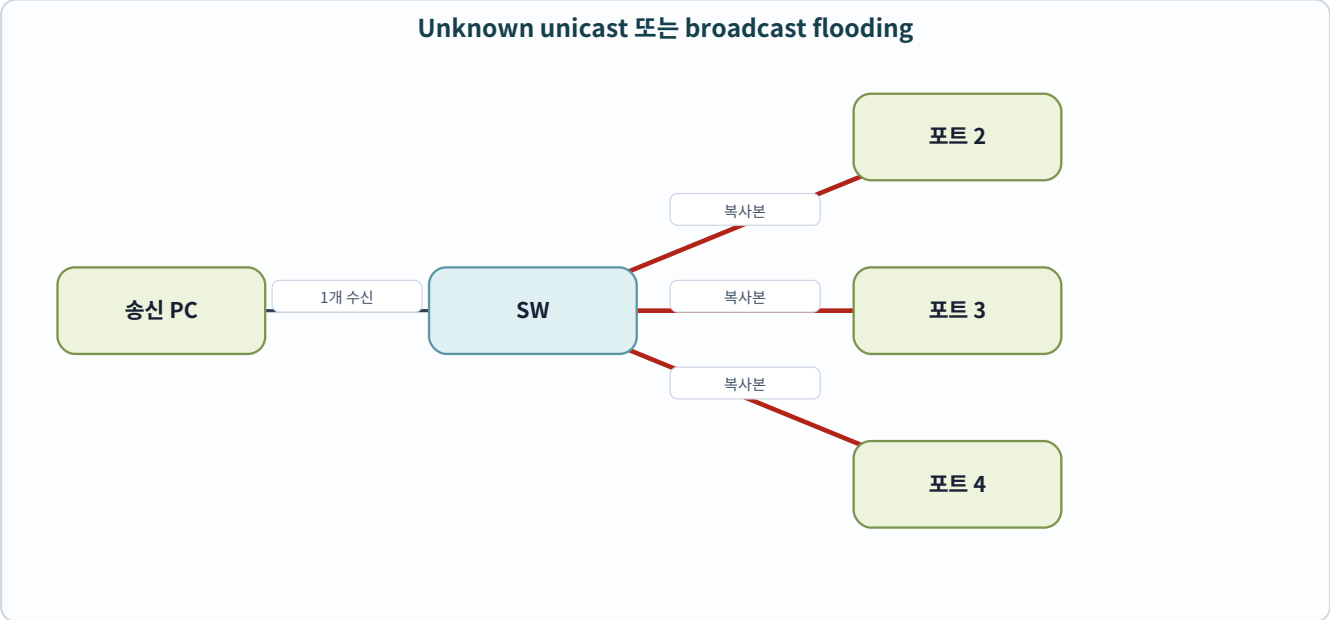


그림 7-3. 스위치가 목적지를 모르면 같은 VLAN의 후보 포트로 복제한다. 이것은 정상 동작이지만 루프가 있으면 폭증의 재료가 된다.

Flooding과 broadcasting의 차이

브로드캐스트는 송신자가 "같은 브로드캐스트 도메인의 모두에게" 보내도록 브로드캐스트 주소를 지정한 트래픽 유형이다. Flooding은 스위치가 전달 방법으로 여러 포트에 복제하는 행위다. Unknown unicast도 flooding될 수 있으므로 두 말을 같은 뜻으로 쓰면 안 된다.

Aging이 필요한 이유

단말이 다른 포트로 이동하거나 꺼졌는데 과거 MAC 항목이 영원히 남으면 프레임이 잘못된 포트로 향한다. 그래서 동적 MAC 항목은 일정 시간 사용되지 않으면 aging out된다. 새 프레임이 다시 들어오면 재학습한다. 정적·sticky secure MAC은 일반 동적 항목과 저장·노화 규칙이 다를 수 있다.

충돌 도메인과 브로드캐스트 도메인

필기 확인 + Ethernet 보강

서로 다른 문제를 나타내는 두 범위

함께 찾을 말: Collision Domain, Broadcast Domain

충돌 도메인과 브로드캐스트 도메인란 충돌 도메인은 공유 매체에서 동시 송신 충돌 가능성을 공유하는 범위이고, 브로드캐스트 도메인은 L2 브로드캐스트가 퍼지는 범위다.

Collision domain

충돌 도메인은 같은 전송 매체를 공유해 두 장치가 동시에 송신하면 신호 충돌이 날 수 있는 범위다. 허브 기반 반이중 Ethernet에서는 허브에 연결된 장치들이 하나의 충돌 도메인을 공유한다. 스위치의 각 포트가 전이중으로 동작하는 현대 Ethernet에서는 포트마다 독립 링크가 되어 CSMA/CD 충돌 개념이 사실상 적용되지 않는다.

Broadcast domain

브로드캐스트 도메인은 L2 브로드캐스트 프레임이 전달되는 범위다. 일반적으로 한 VLAN이 하나의 브로드캐스트 도메인이다. 라우터나 L3 경계는 L2 브로드캐스트를 다른 네트워크로 그대로 전달하지 않는다.

구분	무엇이 경계를 만든다	현대 실무에서 보는 방식
충돌 도메인	스위치 포트/전이중 링크	허브·반이중 환경을 이해할 때 중요. 스위치 포트별로 분리된다.
브로드캐스트 도메인	VLAN 또는 라우터 경계	ARP request, DHCP discover 등의 확산 범위와 장애 영향 범위를 결정한다.

라우터는 왜 브로드캐스트 스톰의 범위를 줄이는가

라우터는 L2 프레임을 그대로 다음 인터페이스로 복제하지 않고, 목적지 IP와 라우팅 표를 보고 새 링크 프레임을 만든다. 따라서 한 VLAN의 Ethernet 브로드캐스트가 기본적으로 다른 라우터 인터페이스까지 넘어가지 않는다.

8장. L2 루프가 생기고 STP가 이를 막는 전 과정

> 중복 회선은 장애 대비에 필요하지만, Ethernet 스위치 사이의 모든 중복 링크를 동시에 전달 상태로 두면 프레임이 원을 그리며 반복될 수 있다. 이 장에서는 “루핑이 무엇이다”라는 한 줄 정의를 넘어서, 첫 브로드캐스트 한 개가 어떻게 복제되고 다시 복제되어 링크와 CPU를 잠식하는지, 왜 MAC 주소 표가 흔들리는지, STP가 어떤 비교 규칙으로 한 길을 논리적으로 막는지 단계별로 따라간다.

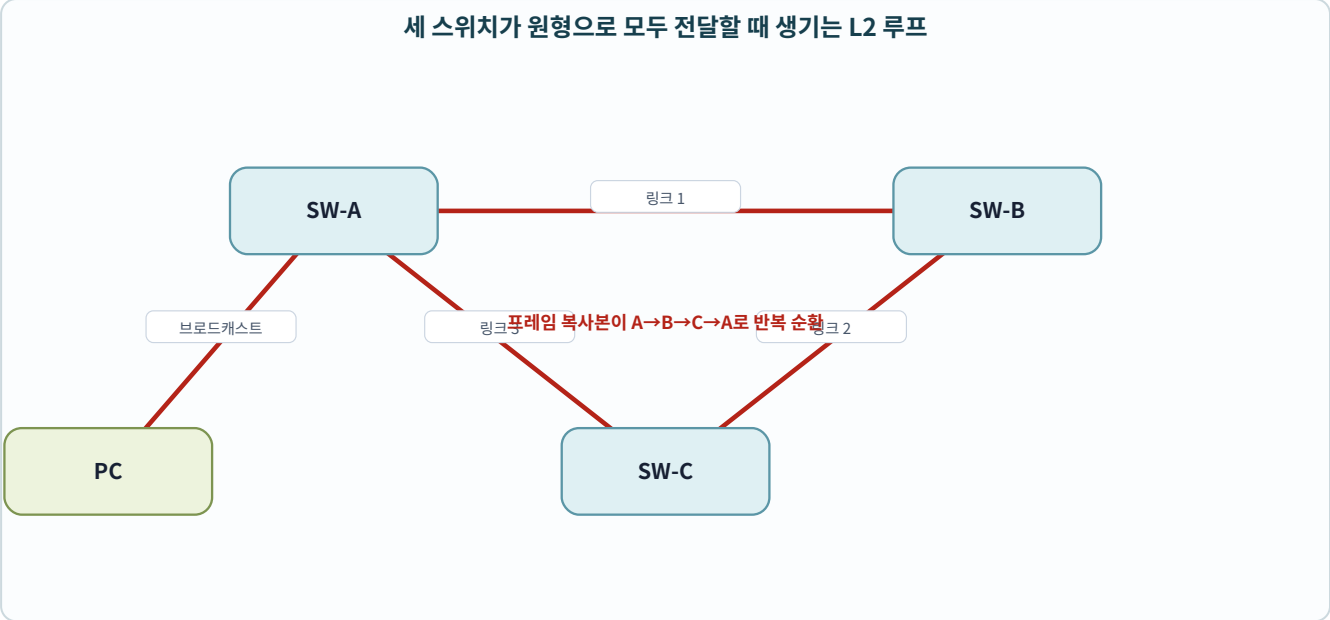


그림 8-1. Ethernet 프레임에는 라우터 홉마다 감소하는 IP TTL과 같은 기본 수명 장치가 없다. 모든 링크가 forwarding이면 복사본이 고리 안을 계속 돈다.

L2 루프와 루핑

필기 확인 + Cisco/IEEE 보강

중복 링크 자체가 아니라 “동시에 열린 논리 전달 경로”가 문제다

함께 찾을 말: Layer 2 Loop, Looping, 루핑

L2 루프와 루핑란 같은 VLAN의 프레임이 둘 이상의 활성 경로를 따라 스위치 사이를 원형으로 반복 순환하고, flooding 때마다 복제되어 네트워크 자원을 소모하는 현상이다.

Loop와 looping의 뜻

Loop는 시작점으로 되돌아오는 닫힌 경로다. **Looping**은 프레임이 그런 경로를 따라 반복 순환하는 현상을 말한다. 케이블이 원형으로 연결되었다고 무조건 장애는 아니다. STP나 다른 제어가 중복 경로 중 하나를 차단해 논리적으로 트리를 만들면 물리적 원형이 있어도 사용자 프레임은 순환하지 않는다. 문제는 같은 VLAN에서 둘 이상의 전달 경로가 동시에 열리는 경우다.

왜 IP 라우팅 루프보다 L2 루프가 급격한가

IP 패킷에는 TTL/Hop Limit이 있어 라우터를 지날 때마다 1씩 줄고 0이 되면 폐기된다. 반면 일반 Ethernet 프레임에는 스위치를 지날 때마다 감소하는 수명 필드가 없다. 또한 스위치는 브로드캐스트와 unknown unicast를 여러 포트로 **복제**하므로 단순 순환을 넘어 복사본 수가 증가할 수 있다.

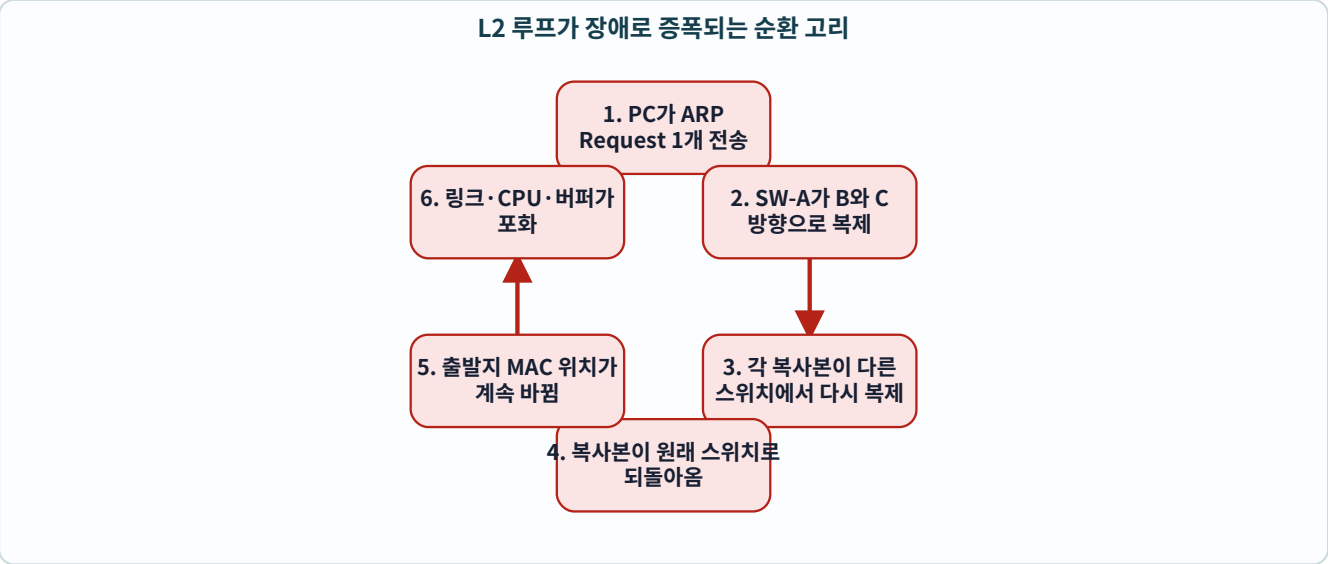


그림 8-2. 프레임 하나가 “돌기만” 하는 것이 아니라, flooding 지점마다 복제되어 수가 늘어난다.

한 프레임이 폭주로 변하는 단계

1. **시작**: PC가 ARP Request 또는 DHCP Discover 같은 브로드캐스트 프레임 하나를 보낸다.
2. **첫 복제**: SW-A는 같은 VLAN의 두 중복 링크로 프레임을 보낸다.
3. **재유입**: SW-B와 SW-C가 받은 프레임을 다른 전달 포트로 다시 flooding한다.

4. **순환**: 복사본이 SW-A로 되돌아오지만 Ethernet 헤더에 "이미 처리했다"는 고유 번호나 TTL이 없어 새 프레임처럼 처리된다.
5. **기하급수적 증가**: 각 스위치가 매번 여러 포트로 복제하면서 동일 내용의 프레임이 폭증한다.
6. **자원 고갈**: 링크 대역폭, 스위치 버퍼, 제어 CPU가 점유되어 정상 프레임이 지연·드롭된다.
7. **제어면 붕괴**: STP BPDU, 관리 SSH, 라우팅 제어 메시지까지 손실되어 복구가 더 어려워진다.

모든 루프가 브로드캐스트 프레임만 돌리는 것은 아니다

브로드캐스트 스톰이라는 표현이 유명하지만, unknown unicast와 일부 멀티캐스트도 루프에서 복제될 수 있다. 또한 이미 알려진 유니캐스트라도 MAC 표가 플래핑하면 잘못된 포트로 보내져 루프·손실이 지속될 수 있다.

물리적·설정상 발생 원인

- 스위치 사이를 두 케이블로 연결했지만 STP가 꺼져 있거나 특정 VLAN에서 BPDU가 막힌 경우
- 사용자가 책상 아래 소형 스위치나 허브로 벽 포트 두 개를 서로 연결한 경우
- EtherChannel 양쪽 설정이 달라 한쪽은 묶음, 다른 쪽은 독립 링크로 인식한 경우
- 가상화 호스트·브리지·vSwitch에서 동일 VLAN이 두 경로로 다시 물리망에 연결된 경우
- 단방향 링크 장애 때문에 BPDU는 못 받지만 데이터는 전달되어 차단 포트가 잘못 열리는 경우
- 트렁크 허용 VLAN/네이티브 VLAN 불일치로 특정 VLAN의 STP 인식만 어긋난 경우

현장에서 보이는 증상

- 인터페이스 입력률·브로드캐스트 비율이 갑자기 치솟는다.
- 동일 MAC 주소가 여러 포트 사이에서 빠르게 이동한다는 MAC move/flapping 로그가 반복된다.
- 스위치 CPU가 높아지고 CLI/SSH 응답이 느려지거나 끊긴다.
- ping 손실과 지연이 불규칙하게 커지며 한 VLAN 전체가 영향을 받는다.
- 포트 LED가 거의 계속 켜져 있거나 매우 빠르게 깜박인다.
- STP topology change 횟수가 빠르게 증가한다.

브로드캐스트 스톰

필기 확인 + Cisco 보강

정상 브로드캐스트가 "폭풍"으로 증폭된 상태

함께 찾을 말: Broadcast Storm, Storm Control

브로드캐스트 스톰란 브로드캐스트 프레임이 비정상적으로 대량 생성되거나 L2 루프에서 반복 복제되어 브로드캐스트 도메인의 링크·버퍼·CPU를 압도하는 상태다.

Storm이라는 표현의 의미

Storm은 짧은 시간에 통제하기 어려운 양의 프레임이 몰아치는 상태를 비유한다. 브로드캐스트 스톰은 브로드캐스트 프레임이 대량으로 생성·복제되어 브로드캐스트 도메인의 대역폭과 장비 처리 능력을 잠식하는 현상이다. 루프가 대표 원인이지만, 오작동 장비나 악성 프로그램이 대량 브로드캐스트를 생성하는 경우도 있다.

정상 브로드캐스트와 스톰의 경계

ARP·DHCP 같은 브로드캐스트는 정상 운영에 필요하다. 문제는 절대 개수 하나로만 판단하지 않는다. 평소 기준선 대비 비율, 인터페이스 용량, CPU·드롭·지연, 발생 VLAN과 송신원 수를 함께 본다. 1Gbps 링크와 저성능 IoT 스위치가 견딜 수 있는 PPS는 다르다.

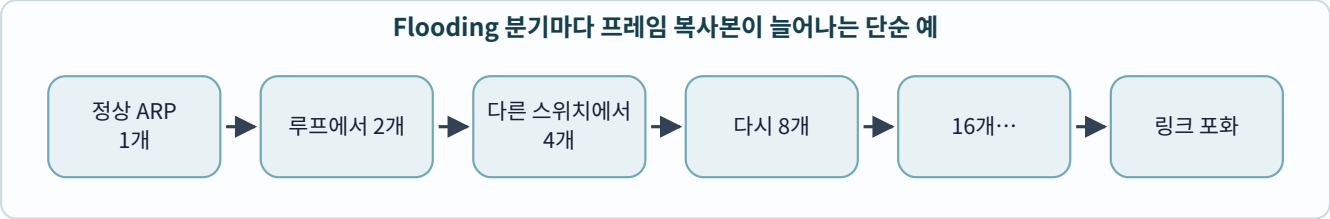


그림 8-3. 실제 증가율은 토폴로지와 포트 수에 따라 달라지지만, 복제-재유입 구조가 폭증의 핵심이다.

Storm control의 역할과 한계

Storm control은 포트에서 브로드캐스트·멀티캐스트·unknown unicast 비율 또는 속도가 임계치를 넘을 때 초과 트래픽을 버리거나 포트를 차단하는 보호 기능이다. 이것은 STP를 대체하지 않는다. 루프의 근본 원인을 제거하지 않은 채 임계치만 낮추면 정상 ARP/DHCP까지 끊길 수 있고, 높이면 장애를 충분히 막지 못한다. 기준선을 측정하고 보호 계층으로 사용해야 한다.

Cisco 계열 storm-control 예시 - 플랫폼별 문법·단위 확인 필요

```
interface GigabitEthernet0/10
 storm-control broadcast level 1.00 0.50
 storm-control multicast level 2.00 1.00
 storm-control action trap

show storm-control
show interfaces counters errors
```

MAC 플래핑

필기 확인 + Cisco 보강

루프의 대표 증상이지만 원인은 여러 가지
다

함께 찾을 말: MAC Flapping, MAC Move

MAC 플래핑란 같은 VLAN의 동일 출발지 MAC 주소가 짧은 시간 동안 서로 다른 스위치 포트에서 반복 관찰되어 MAC 주소 표의 위치가 계속 바뀌는 현상이다.

MAC flapping의 정확한 의미

스위치는 프레임의 출발지 MAC을 수신 포트에 연결해 학습한다. 같은 VLAN의 동일 MAC 주소가 매우 짧은 간격으로 포트 A, 포트 B, 다시 포트 A에서 보이면 표의 위치가 계속 바뀐다. 이 현상을 MAC flapping 또는 MAC move라고 한다.

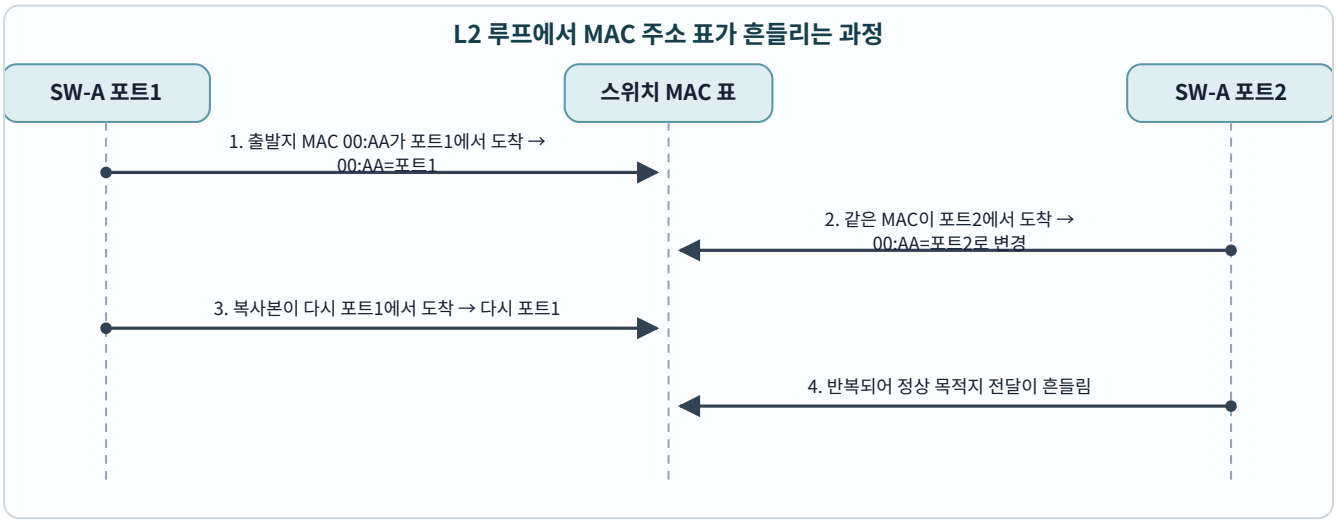


그림 8-4. 스위치는 “최신 출발지 위치”를 믿기 때문에 루프 복사본이 들어오는 방향에 따라 항목이 계속 바뀐다.

루프 외의 원인

- VM live migration 또는 active-active NIC teaming처럼 정상적으로 MAC 위치가 바뀌는 설계
- vPC/MLAG 같은 멀티섀시 구성에서 제어면·peer-link 이상
- 같은 MAC 주소를 복제한 장비나 가상 머신
- EtherChannel 멤버 불일치
- MAC spoofing 또는 잘못 연결된 브리지

로그 한 줄로 공격이라고 단정하지 않기

MAC flapping은 증상이다. 발생 VLAN, 두 포트의 물리 경로, STP 상태, EtherChannel 구성, 가상화 이동 기록을 함께 확인해 원인을 분리한다.

STP

필기 확인 + IEEE/Cisco 검증

이중화는 남기고 전달 경로만 트리로 제한한다

함께 찾을 말: Spanning Tree Protocol, 스패닝 트리

STP란 중복된 Ethernet 스위치 경로에서 루프가 없는 논리 트리를 계산하고 일부 포트를 차단하여 L2 프레임 순환을 막는 프로토콜이다.

Spanning Tree라는 이름

Spanning은 모든 정점을 빠짐없이 연결한다는 뜻이고, **Tree**는 순환이 없는 연결 구조다. STP는 모든 스위치가 연결된 상태는 유지하면서, L2 루프가 생기지 않도록 일부 포트를 사용자 프레임 비전달 상태로 만든다.

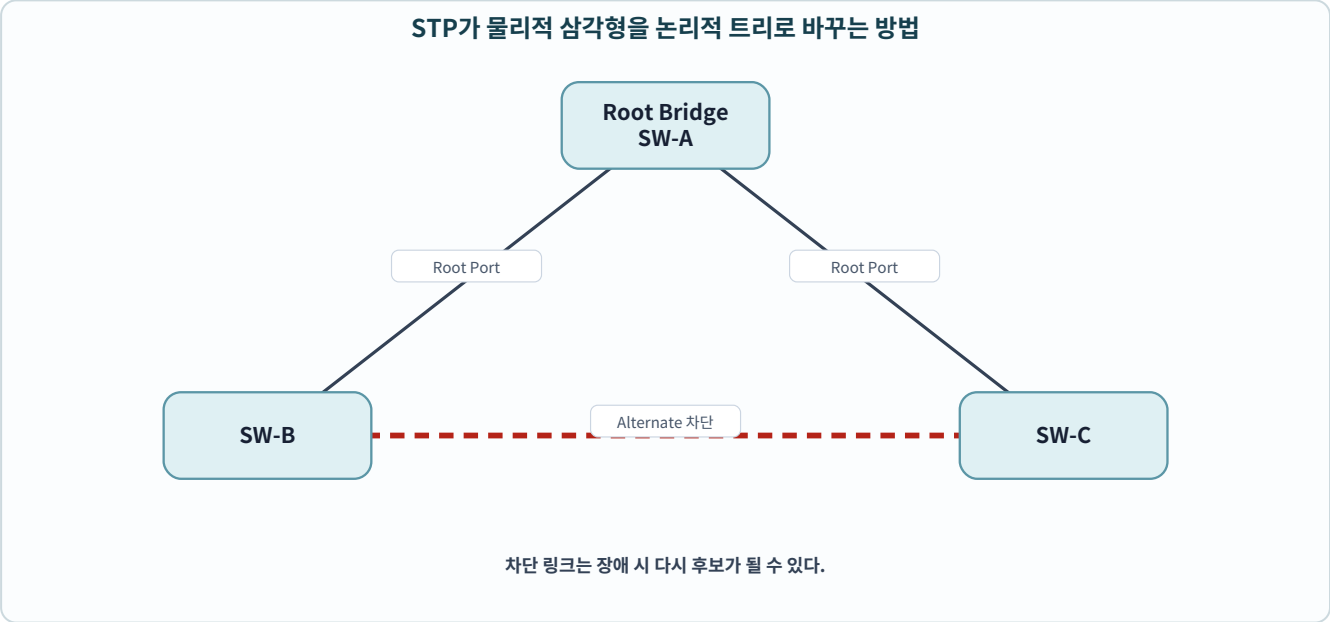


그림 8-5. 중복 케이블을 물리적으로 제거하지 않고 한 경로를 논리적으로 차단한다. 주 경로 장애 시 차단 경로를 전환할 수 있다.

STP가 하는 일을 한 문장으로

스위치들이 BPDU라는 제어 메시지를 주고받아 하나의 루트 브리지를 선출하고, 각 스위치에서 루트까지의 최선 경로와 각 세그먼트의 대표 전달 포트를 정한 뒤, 나머지 중복 포트를 차단한다.

동작 전체 순서

1. 각 스위치는 처음에 자신이 루트라고 가정한 BPDU를 보낸다.
2. 더 낮은 Bridge ID를 가진 루트 정보를 받으면 그 정보를 더 우수하다고 인정한다.
3. 루트가 아닌 각 스위치는 루트까지 총 경로 비용이 가장 낮은 포트를 **Root Port**로 선택한다.
4. 각 LAN 세그먼트에서는 루트까지 더 좋은 경로를 제공하는 포트가 **Designated Port**가 된다.
5. 그 외 중복 포트는 Alternate/Blocking 계열 역할로 남아 사용자 프레임 전달을 멈춘다.

6. 링크 장애나 BPDU 변화가 생기면 토폴로지를 다시 계산하고 필요할 때 대체 포트를 forwarding으로 전환한다.

STP는 대역폭 묶음 기능이 아니다

STP는 루프를 막기 위해 중복 경로 일부를 차단한다. 여러 물리 링크를 하나의 논리 링크로 동시에 사용하려면 EtherChannel/LAG 같은 기술이 필요하다. 단순히 케이블을 두 개 꽂았다고 대역폭이 두 배가 되지 않는다.

BPDU

필기 확인 + IEEE/Cisco 보강

STP의 판단 재료를 운반하는 제어 프레임

함께 찾을 말: Bridge Protocol Data Unit, Superior BPDU

BPDU란 스위치들이 STP의 루트, 경로 비용, 브리지·포트 식별자와 타이머를 교환하여 루프 없는 토폴로지를 계산하는 제어 메시지다.

BPDU란 무엇인가

BPDU(Bridge Protocol Data Unit)는 스위치가 STP 계산에 필요한 정보를 서로 알리는 L2 제어 프레임이다. 일반 사용자 데이터가 아니라 "누가 루트인지, 루트까지 비용이 얼마인지, 이 메시지를 누가 어느 포트에서 보냈는지, 타이머가 얼마인지"를 담는다.

Configuration BPDU에서 비교하는 핵심 정보

Protocol ID	Version/Type	Flags	Root ID	Root Path Cost	Bridge ID	Port ID	Timers
STP 식별	STP/RSTP	TC 등	루트 후보	누적 비용	송신 스위치	송신 포트	Hello 등

그림 8-6. 구현·버전에 따라 세부 형식은 다르지만, 루트 정보와 비용, 송신자·포트, 타이머가 핵심이다.

Superior BPDU의 비교 순서

1. 더 낮은 **Root Bridge ID**
2. 같은 루트라면 더 낮은 **Root Path Cost**
3. 그래도 같으면 더 낮은 **Sender Bridge ID**
4. 그래도 같으면 더 낮은 **Sender Port ID**

이 비교를 튜플처럼 왼쪽부터 차례로 생각하면 된다. 첫 항목에서 승부가 나면 뒤 항목은 보지 않는다. "속도가 빠른 링크가 무조건 이긴다"가 아니라 먼저 같은 루트 정보를 비교하고, 그 안에서 경로 비용을 본다.

BPDU가 사라질 때 왜 위험한가

차단 포트는 상대에게서 더 우수한 BPDU를 계속 받기 때문에 자신이 중복 경로임을 안다. 단방향 장애나 필터링으로 BPDU만 끊기면, 포트가 “이제 내가 지정 포트여야 한다”고 오판해 forwarding으로 열릴 수 있다. Loop Guard, UDLD 같은 보호 기능이 이런 상황을 보완한다.

루트 브리지와 Bridge ID

필기 확인 + Cisco 보강

트리의 기준점을 우연에 맡기지 않는다

함께 찾을 말: Root Bridge, Bridge ID, BID, Root Election

루트 브리지와 Bridge ID란 STP 토폴로지의 기준점이 되는 스위치와, 그 루트를 선출하기 위해 우선순위·식별 정보를 결합한 비교 값이다.

Bridge ID

Bridge ID는 STP에서 스위치의 우선순위를 비교하는 식별 값이다. 전통적으로 Bridge Priority와 MAC 주소가 결합된다. VLAN별 STP 구현에서는 Extended System ID가 VLAN 정보를 반영한다. 가장 낮은 Bridge ID가 Root Bridge가 된다.

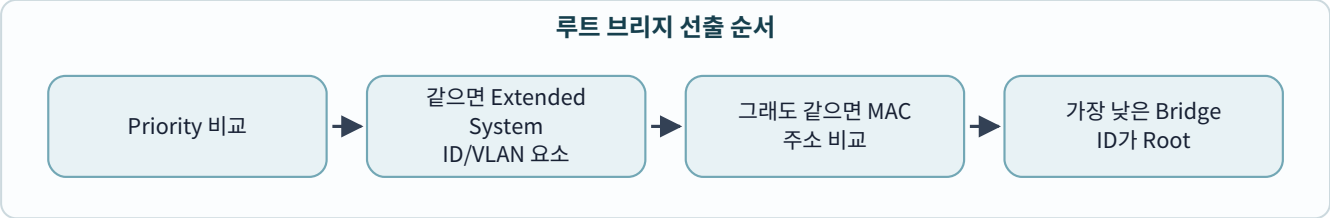


그림 8-7. 기본 우선순위만 두면 MAC 주소 값에 따라 우연히 루트가 정해질 수 있다.

왜 루트를 설계해야 하는가

루트 브리지는 트리의 기준점이다. 액세스 계층의 저성능 스위치가 우연히 루트가 되면 트래픽 경로가 비효율적으로 굽고, 장애 시 수렴도 예상과 달라진다. 일반적으로 네트워크 중심의 배포/코어 스위치를 primary/secondary root로 의도적으로 정한다.

루트 브리지 설정·검증 예시

```
# 예시 – 실제 VLAN/플랫폼 정책에 맞춰 사용
spanning-tree vlan 10 root primary
spanning-tree vlan 10 root secondary

show spanning-tree vlan 10
show spanning-tree root
```

낮을수록 우수하다

STP의 Bridge ID, 경로 비용, Port ID 비교는 기본적으로 “값이 낮을수록 더 우수”하다. ‘우선순위가 높다’는 일상 표현과 숫자 방향이 반대여서 혼동하기 쉽다.

STP 포트 역할

필기 확인 + IEEE/Cisco 보강

역할과 상태를 구분해야 한다

함께 찾을 말: Root Port, Designated Port, Alternate Port, Blocking Port

STP 포트 역할란 루트 방향의 최선 포트, 세그먼트의 대표 포트, 중복 대체 포트처럼 STP 계산 결과 각 포트에 부여되는 토폴로지상의 역할이다.

Root Port

루트가 아닌 각 스위치에서 루트 브리지까지 가장 우수한 경로를 제공하는 포트 하나다. 해당 스위치가 루트 방향으로 보낼 때 사용하는 대표 포트다. 루트 브리지 자체에는 Root Port가 없다.

Designated Port

각 LAN 세그먼트에서 그 세그먼트의 프레임을 루트 방향으로 가장 잘 전달할 포트다. 루트 브리지의 활성 포트는 일반적으로 Designated Port가 된다.

Alternate/Blocking

같은 목적을 위한 더 나은 포트가 이미 있으므로 루프를 막기 위해 사용자 프레임을 전달하지 않는 대체 포트다. 물리 링크는 살아 있고 BPDU는 처리할 수 있으며, 주 경로 장애 시 전환 후보가 된다.

역할	선택 범위	사용자 프레임	핵심 질문
Root Port	비루트 스위치마다 1개	Forwarding	이 스위치가 루트로 가는 최선 입구는?
Designated Port	세그먼트마다 1개	Forwarding	이 세그먼트에서 루트 방향 대표는?
Alternate/Blocking	중복 경로	차단	이미 더 좋은 경로가 있으므로 어느 포트를 막을까?

포트 역할과 포트 상태는 다른 축이다

Root/Designated/Alternate는 토폴로지에서의 **역할**이고, Discarding/Learning/Forwarding은 현재 데이터 처리 **상태**다. 용어를 섞으면 RSTP 전환 과정을 설명하기 어렵다.

STP 포트 상태와 RSTP

필기 확인 + IEEE/Cisco 보강

루프 없이 안전하게 전달 상태로 전환한다

함께 찾을 말: Blocking, Listening, Learning, Forwarding, RSTP, 802.1w

STP 포트 상태와 RSTP란 STP 포트가 사용자 프레임을 버리고, MAC을 학습하고, 최종 전달하기까지 거치는 상태와 이를 더 빠르게 수렴하도록 개선한 프로토콜이다.



그림 8-8. 포트는 곧바로 사용자 프레임을 전달하지 않고 토폴로지 안정화와 MAC 학습 단계를 거친다.

전통 802.1D 상태

Blocking

사용자 프레임을 전달·학습하지 않고 BPDU를 듣는다.

Listening

STP 계산에 참여하지만 아직 MAC 학습과 사용자 전달을 하지 않는다.

Learning

출발지 MAC은 학습하지만 사용자 프레임은 아직 전달하지 않는다.

Forwarding

MAC 학습과 사용자 프레임 전달을 수행한다.

Disabled

관리적 shutdown 또는 링크 비활성 등으로 STP 참여하지 않는다.

RSTP가 빨라지는 이유

RSTP(802.1w)는 상태를 Discarding, Learning, Forwarding으로 단순화하고, 점대점 링크에서 proposal/agreement handshake로 양쪽이 안전하게 동기화되면 전통 타이머를 모두 기다리지 않고 빠르게 전환한다. Edge port는 단말 연결로 간주해 즉시 forwarding할 수 있다.

빠른 수렴도 잘못된 경계 설정 앞에서는 위험하다

스위치가 연결된 포트를 edge/PortFast로 잘못 지정하면 루프가 생길 때 즉시 forwarding할 수 있다. 그래서 단말 포트에는 BPDU Guard를 함께 적용해 스위치 연결을 탐지하는 구성이 흔하다.

STP 보호 기능

필기 보강 + Cisco 검증

각 기능의 위협 모델과 적용 위치가 다르다

함께 찾을 말: PortFast, BPDU Guard, Root Guard, Loop Guard, UDLD

STP 보호 기능란 STP의 정상 계산만으로 처리하기 어려운 잘못된 스위치 연결, 예상치 못한 루트 변경, BPDU 단절·단방향 장애를 탐지하고 차단하는 보조 기능들이다.

기능	무엇을 막거나 빠르게 하는가	일반 적용 위치
PortFast / Edge	단말 포트의 불필요한 STP 대기 시간을 줄임	PC·서버 등 최종 단말 포트
BPDU Guard	단말 포트에서 BPDU를 받으면 포트를 err-disable 등으로 차단	PortFast/edge 포트
Root Guard	특정 방향에서 더 우수한 BPDU가 들어와 루트가 바뀌는 것을 막음	하위/비신뢰 스위치 방향
Loop Guard	받아야 할 BPDU가 사라져 차단 포트가 잘못 열리는 상황을 완화	스위치 간 중복 링크
UDLD	광 링크 등의 단방향 통신 장애를 탐지	스위치 간 링크

PortFast는 STP를 끄는 기능이 아니다

PortFast 포트도 BPDU를 송수신할 수 있고 STP 도메인에 존재한다. 단지 단말이 연결된 것으로 가정해 Listening/Learning 지연을 건너뛰고 빠르게 forwarding한다. 그래서 그 가정이 깨져 스위치가 연결되면 위험하며 BPDU Guard가 보완한다.

단말 포트 보호 예시

```
interface GigabitEthernet0/10
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable

show spanning-tree interface gigabitEthernet0/10 detail
show errdisable recovery
```

루프 장애 대응 순서

1. 변경·케이블 연결 직후인지 확인하고 영향 VLAN을 특정한다.
2. 코어에서 무작정 재부팅하지 말고, MAC move 로그와 인터페이스 broadcast PPS를 본다.
3. STP root, blocked/alternate 포트, topology change 횟수를 확인한다.
4. 최근 연결된 액세스 포트 또는 비정상적으로 높은 입력 포트를 단계적으로 차단해 폭주 범위를 줄인다.
5. EtherChannel · trunk · native VLAN · 가상 브리지 구성을 양쪽에서 대조한다.
6. 복구 후 BPDU Guard, Loop Guard, storm-control, 변경 절차를 보완한다.

9장. VLAN·802.1Q·포트 시큐리티: 논리 경계와 접속 통제

VLAN은 하나의 스위치를 여러 논리 LAN으로 나누고, 802.1Q는 여러 VLAN 프레임이 같은 물리 링크를 지날 때 소속을 표시한다. 포트 시큐리티는 그 포트에 나타날 수 있는 출발지 MAC 주소와 개수를 제한한다. 세 기술은 모두 스위치 설정에 등장하지만 해결하는 문제가 다르다. 이 장은 "VLAN 10"이라는 번호가 프레임 처리에서 실제로 어떤 의미가 되는지부터, `switchport port-security mac-address sticky`가 어떤 상태를 만드는지까지 설명한다.

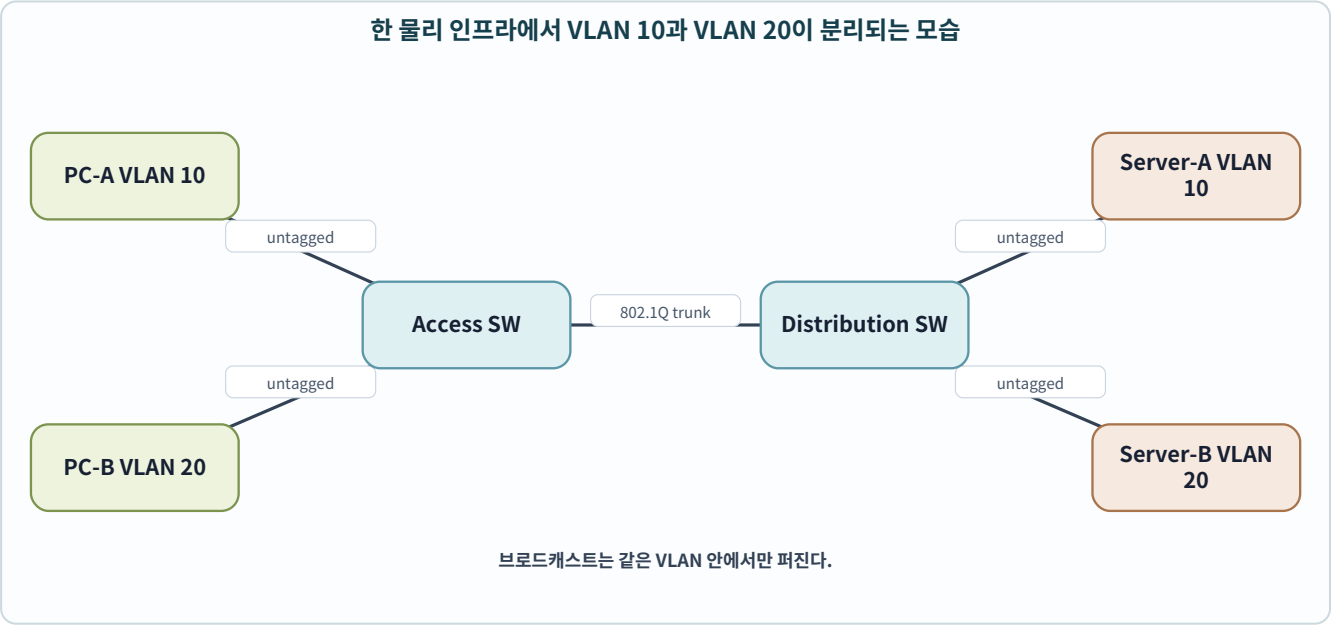


그림 9-1. 액세스 포트의 단말은 보통 태그 없는 프레임을 보내고, 스위치 간 트렁크에서 802.1Q 태그로 VLAN을 구분한다.

VLAN

필기 확인 + IEEE/Cisco 보강

물리 배치와 논리 네트워크 경계를 분리한다

함께 찾을 말: Virtual LAN, 가상 LAN, LAN과 VLAN

VLAN란 하나의 물리적 스위칭 인프라를 여러 독립적인 L2 브로드캐스트 도메인으로 논리 분할하는 기술이다.

Virtual이라는 말의 의미

VLAN의 “Virtual”은 가짜라는 뜻이 아니라, 물리 케이블 배치와 별개로 설정을 통해 논리적인 LAN 경계를 만든다는 뜻이다. 같은 스위치에 꽂힌 두 PC도 서로 다른 VLAN이면 L2 브로드캐스트를 직접 주고받지 못하고, 다른 층의 스위치에 있는 PC도 트렁크로 같은 VLAN을 연장하면 같은 브로드캐스트 도메인에 속할 수 있다.

스위치 내부에서 VLAN이 붙는 순간

1. 태그 없는 프레임이 액세스 포트로 들어온다.
2. 스위치는 포트 설정을 보고 내부적으로 VLAN ID를 부여한다.
3. MAC 학습도 단순 MAC이 아니라 **VLAN + MAC** 문맥에서 이뤄진다.
4. 목적지 조회와 flooding은 같은 VLAN 범위 안에서만 수행된다.
5. 다른 스위치로 트렁크를 통해 보낼 때는 802.1Q 태그를 삽입해 VLAN 소속을 전달한다.
6. 최종 액세스 포트로 나갈 때는 일반적으로 태그를 제거해 단말에 전달한다.

VLAN의 효과와 한계

- 브로드캐스트 범위와 장애 영향 범위를 분리한다.
- 부서·서비스·신뢰 수준별로 L2 경계를 만든다.
- VLAN 간 트래픽을 L3 장비·방화벽을 통과시켜 정책을 적용할 수 있다.
- 그러나 VLAN만으로 사용자 인증, 암호화, 완전한 공격 차단이 되는 것은 아니다.

VLAN 번호가 IP 네트워크를 자동 생성하지 않는다

VLAN은 L2 경계이고 IP subnet은 L3 주소 범위다. 운영에서는 보통 VLAN 10에 10.10.10.0/24처럼 1:1로 매핑하지만, 스위치가 VLAN을 만들었다고 IP 주소와 게이트웨이가 자동으로 생기는 것은 아니다.

Access port와 Trunk port

필기 확인 + Cisco 검증

포트 역할에 따라 VLAN을 부여하는 방법이 달라진다

함께 찾을 말: 액세스 포트, 트렁크 포트, Uplink

> **Access port와 Trunk port란** Access port는 보통 하나의 VLAN에 태그 없는 단말 프레임을 연결하고, Trunk port는 802.1Q 태그로 여러 VLAN 프레임을 한 링크에서 구분해 운반한다.

Access port란

Access port는 일반적으로 하나의 액세스 VLAN에 단말을 연결하는 스위치 포트다. 단말이 태그 없는 Ethernet 프레임을 보내면, 스위치는 포트에 설정된 VLAN으로 분류한다. 나갈 때도 보통 태그를 제거한다.

Trunk port란

Trunk port는 여러 VLAN의 프레임을 하나의 물리 링크에서 운반하는 포트다. 스위치-스위치, 스위치-라우터, 스위치-가상화 호스트 사이에 사용한다. 프레임마다 VLAN 소속을 구분해야 하므로 802.1Q 태그를 사용한다.

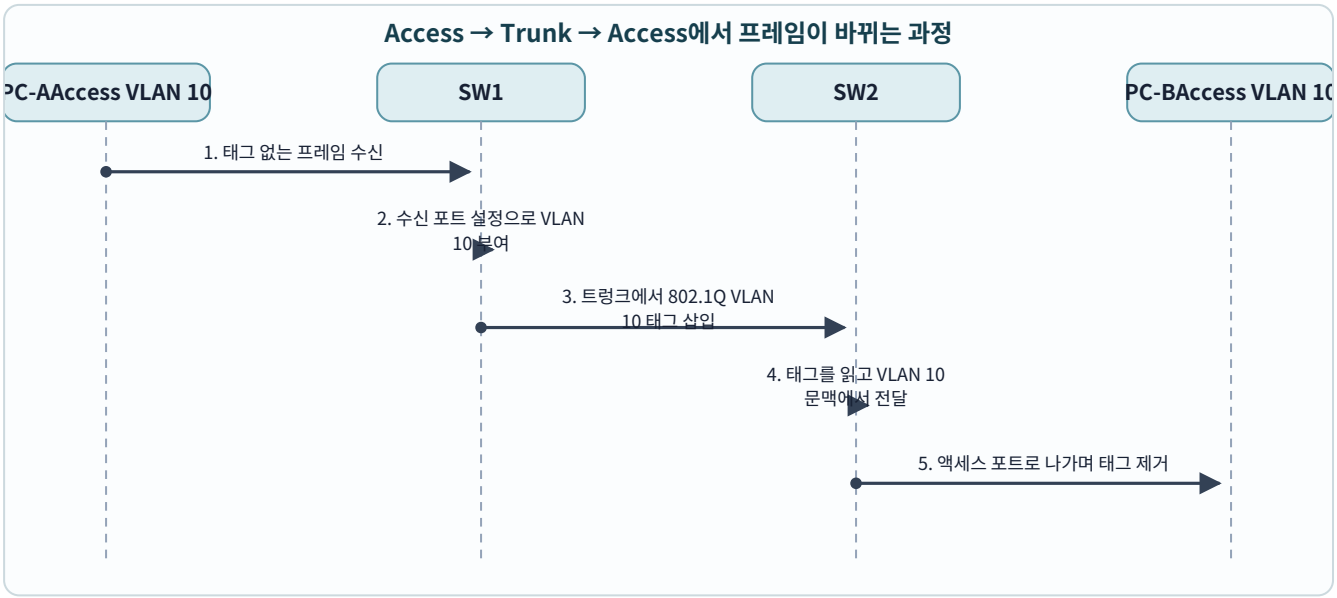


그림 9-2. 태그는 종단 PC가 반드시 보는 정보가 아니라, 중간 스위치가 공유 링크에서 VLAN을 구분하기 위한 정보다.

항목	Access port	Trunk port
주 목적	한 VLAN의 일반 단말 연결	여러 VLAN을 링크 하나로 운반
일반 프레임	태그 없음	대부분 802.1Q 태그 있음
VLAN 선택	포트의 access VLAN 설정	프레임 태그와 native VLAN 규칙
대표 연결	PC, 프린터, 서버 NIC	스위치, 라우터, 하이퍼바이저

Access/Trunk 명시 설정 예시

```
interface GigabitEthernet0/10
 description USER-PC
 switchport mode access
 switchport access vlan 20
 spanning-tree portfast
 spanning-tree bpduguard enable

interface GigabitEthernet0/1
 description UPLINK-TO-SW2
 switchport mode trunk
 switchport trunk allowed vlan 10,20,30,999
 switchport trunk native vlan 999
 switchport nonegotiate
```

IEEE 802.1Q (dot1q)

필기 확인 + IEEE/Cisco/RFC 검증

"프레임이 어느 VLAN 소속인가"를 링크 건너편에 전달한다

함께 찾을 말: dot1q, 802.1Q, VLAN tagging, 태깅, TPID, PCP, DEI, VID

IEEE 802.1Q (dot1q)란 여러 VLAN이 같은 Ethernet 링크를 공유할 때 프레임에 4바이트 태그를 삽입하여 VLAN 소속과 우선순위 정보를 표시하는 브리징·VLAN 표준이다.

Dot1Q라는 이름

dot1q는 "IEEE 802.1Q"를 읽기 쉽게 부르는 이름이다. IEEE 802.1Q는 단순히 Cisco 명령어 한 줄이 아니라, 브리지와 VLAN 동작 및 Ethernet 프레임의 VLAN 태깅 규칙을 정의하는 표준이다.

태그가 왜 필요한가

스위치 간 링크 하나가 VLAN 10, 20, 30의 프레임을 모두 운반한다고 하자. 태그가 없으면 수신 스위치는 프레임이 어느 VLAN에서 왔는지 구분할 수 없다. 802.1Q는 출발지 MAC 뒤에 4바이트 태그를 삽입해 VLAN 소속과 우선순위 정보를 전달한다.

802.1Q 태그가 삽입된 Ethernet 프레임



그림 9-3. TPID 0x8100이 태그 존재를 알리고, TCI 안의 PCP·DEI·VID가 우선순위, 드롭 가능성, VLAN ID를 표현한다. 태그 삽입으로 프레임 내용이 바뀌므로 FCS도 다시 계산한다.

4바이트 태그의 필드

TPID (Tag Protocol Identifier)

보통 0x8100으로, 뒤에 802.1Q 태그 제어 정보가 온다는 것을 나타내는 16비트 값이다.

PCP (Priority Code Point)

3비트 우선순위 값으로 802.1p/QoS 분류에 활용된다. VLAN 번호와 다른 필드다.

DEI (Drop Eligible Indicator)

혼잡 시 상대적으로 먼저 버릴 수 있는 프레임을 표시하는 1비트다.

VID (VLAN Identifier)

12비트 VLAN 식별자다. 0과 4095는 특별/예약 의미가 있어 일반 VLAN으로 쓰지 않으며 흔히 1~4094 범위를 말한다.

프레임 길이와 MTU 오해

802.1Q 태그가 4바이트 추가되므로 링크와 장비는 태그된 Ethernet 프레임 크기를 처리할 수 있어야 한다. IP MTU 1500과 Ethernet 프레임 전체 길이는 같은 값이 아니다. NIC·스위치가 표준 VLAN 태그를 지원하는 환경에서는 이를 정상 처리하지만, 터널·Q-in-Q까지 겹치면 MTU 설계를 확인해야 한다.

802.1Q 태그는 “IP 패킷 안”에 들어가지 않는다

태그는 Ethernet 헤더에 삽입되는 L2 정보다. IP 패킷은 그 뒤 Payload에 들어간다. 라우터가 L2 프레임을 벗기고 새 링크 프레임을 만들면 태그도 나가는 인터페이스 설정에 따라 새로 붙거나 제거된다.

802.1Q VLAN 매핑 예시

```
# Cisco router-on-a-stick 예시
interface GigabitEthernet0/0.10
 encapsulation dot1Q 10
 ip address 10.10.10.1 255.255.255.0

# Linux VLAN 인터페이스 예시
ip link add link eth0 name eth0.10 type vlan id 10
ip addr add 10.10.10.2/24 dev eth0.10
ip link set eth0.10 up
```

Native VLAN

필기 확인 + Cisco 보강

VLAN 1과 같은 말이 아니며 양쪽 설정이 반드시 맞아야 한다

함께 찾을 말: 네이티브 VLAN, Native VLAN mismatch, VLAN 1

Native VLAN란 802.1Q 트렁크에서 태그 없이 들어오거나 기본적으로 태그 없이 나가는 프레임을 분류하는 VLAN 역할이다.

Native VLAN의 정확한 뜻

802.1Q 트렁크에서 **태그가 없는 프레임을 어느 VLAN으로 취급할지** 정한 VLAN이다. Cisco의 전통적 기본값이 VLAN 1인 경우가 많아 “native VLAN = VLAN 1”로 외우기 쉽지만, native VLAN은 역할이고 VLAN 1은 번호다. 설정으로 999 같은 다른 VLAN을 지정할 수 있다.

송신과 수신에서의 처리

1. 송신 측은 기본 동작상 native VLAN 프레임을 태그 없이 내보낼 수 있다.
2. 수신 측은 트렁크에서 태그 없이 들어온 프레임을 자신의 native VLAN에 넣는다.
3. 양쪽 native VLAN이 다르면 같은 무태그 프레임을 서로 다른 VLAN으로 해석한다.
4. 그 결과 CDP/STP 경고, 통신 혼선, VLAN hopping 위험 조건이 생길 수 있다.

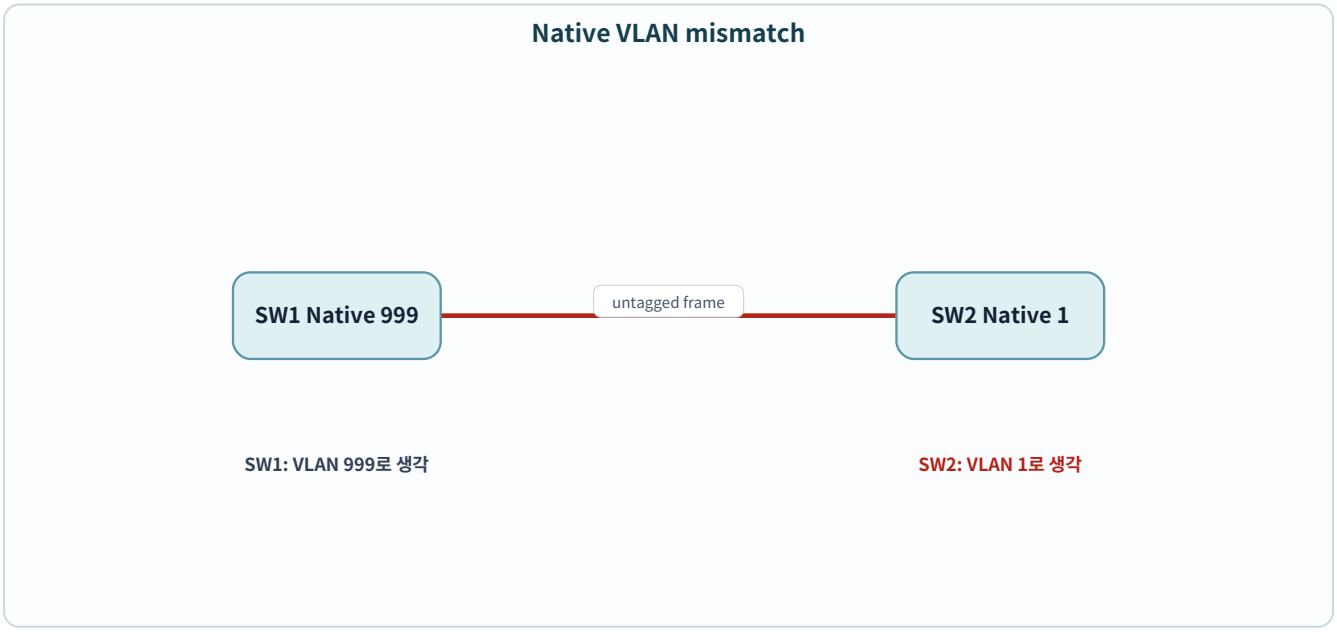


그림 9-4. 같은 태그 없는 프레임을 양쪽이 다른 VLAN으로 분류한다.

운영 원칙

- 사용자 데이터가 없는 별도 native VLAN을 정하고 양쪽을 일치시킨다.
- 필요 없는 VLAN은 trunk allowed list에서 제거한다.
- 가능한 플랫폼에서는 native VLAN도 태그하도록 정책을 검토한다.
- DTP를 끄고 access/trunk 역할을 명시한다.

DTP

필기 확인 + Cisco 보강

편리하지만 포트 역할을 모호하게 만들 수 있다

함께 찾을 말: Dynamic Trunking Protocol, dynamic auto, dynamic desirable

---

DTP란 Cisco 스위치 포트 사이에서 트렁크 형성 여부와 캡슐화 동작을 동적으로 협상하는 L2 제어 기능이다.

DTP란

DTP(Dynamic Trunking Protocol)는 Cisco 스위치 포트끼리 access/trunk 역할을 협상하는 기능이다. dynamic desirable은 적극적으로 trunk를 제안하고, dynamic auto는 상대가 제안하면 trunk가 될 수 있다.

왜 편리함이 위험이 될 수 있는가

사용자 포트를 자동 협상 상태로 두면 예상치 못한 스위치나 공격 장치와 trunk가 형성될 가능성이 생긴다. 따라서 역할이 정해진 포트는 access 또는 trunk로 명시하고, 필요하면 `switchport nonegotiate`로 DTP를 끈다.

로컬 모드	상대 모드 예	결과 개념
access	무관	access로 고정
trunk	trunk/협상 가능	trunk로 고정
dynamic desirable	auto/desirable/trunk	trunk 형성 가능
dynamic auto	desirable/trunk	상대가 적극적이면 trunk

VTP

필기 확인 + Cisco 보강

Trunking 자체가 아니라 VLAN 설정 배포를 다룬다

함께 찾을 말: VLAN Trunking Protocol, VTP Server, VTP Client, VTP Transparent, Configuration Revision, VTP Pruning

> **VTP란** 같은 Cisco VTP 도메인의 스위치 사이에서 VLAN 데이터베이스 변경 정보를 광고·동기화하는 프로토콜이다.

VTP가 해결하려던 문제

스위치가 많을 때 VLAN 생성·삭제를 장비마다 반복하면 오타와 누락이 생긴다. VTP(VLAN Trunking Protocol)는 같은 VTP domain의 Cisco 스위치 사이에 VLAN 데이터베이스 변경을 광고한다.

모드와 Revision

Server

VLAN을 생성·변경하고 광고할 수 있는 전통적 중심 모드다.

Client

광고받은 VLAN 정보를 사용하고 로컬 변경이 제한되는 모드다.

Transparent

자기 VLAN을 로컬에서 관리하고 VTP 메시지를 통과시키는 방식이다. 버전별 세부 동작을 확인한다.

Configuration Revision

VLAN 데이터베이스 변경 세대 번호다. 과거 VTP에서 높은 revision의 잘못된 데이터베이스가 정상망을 덮는 사고가 발생할 수 있었다.

VTP pruning

Pruning은 특정 VLAN이 필요 없는 trunk 방향으로 그 VLAN의 flooded traffic을 보내지 않도록 줄이는 기능이다. “VLAN 우선순위를 정하는 기능”이 아니다.

현대 운영에서는 중앙 자동화와 명시적 관리도 고려한다

VTP 버전과 장비 정책에 따라 안전 장치가 개선되었지만, VLAN 변경의 영향이 크므로 많은 조직은 transparent/off 또는 구성 관리·자동화 도구로 명시적으로 관리한다. 새 스위치를 연결하기 전에 domain, mode, version, revision, VLAN database를 확인한다.

Inter-VLAN routing

필기 확인 + Cisco 보강

VLAN 경계를 넘을 때 L2 프레임은 새로 만들어진다

함께 찾을 말: Router-on-a-stick, Subinterface, SVI, L3 Switch

Inter-VLAN routing란 서로 다른 VLAN·IP subnet 사이의 패킷을 라우터 또는 L3 스위치가 L3 라우팅하고, 각 VLAN에 맞는 새 L2 프레임으로 전달하는 과정이다.

왜 같은 스위치 안에서도 라우터가 필요한가

VLAN 10과 VLAN 20은 서로 다른 L2 브로드캐스트 도메인이다. PC-A는 목적지 IP에 자신의 subnet mask를 적용해 직접 연결된 네트워크가 아니라고 판단하고 default gateway로 보낸다. L3 장비가 IP 패킷을 라우팅한 뒤 VLAN 20의 새 Ethernet 프레임으로 캡슐화한다.

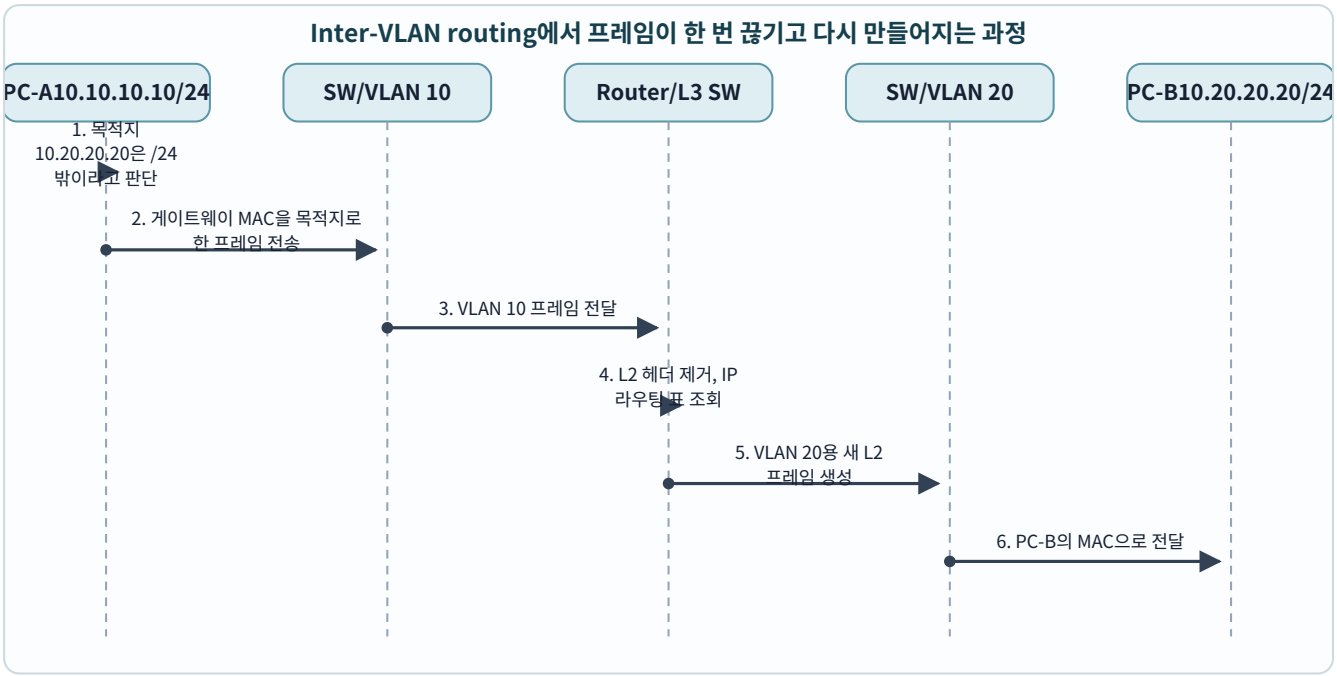


그림 9-5. VLAN 경계를 넘을 때 기존 Ethernet 프레임은 종료되고, 라우터/L3 스위치가 목적지 VLAN에 맞는 새 프레임을 만든다.

대표 구현 3가지

물리 인터페이스별 라우팅

라우터의 물리 포트를 VLAN별 access port에 연결한다. 단순하지만 포트가 많이 필요하다.

Router-on-a-stick

라우터 한 물리 인터페이스를 trunk로 연결하고 VLAN별 subinterface와 802.1Q encapsulation을 구성한다.

L3 switch SVI

스위치에 VLAN별 SVI(Vlan10 등)를 만들고 하드웨어 기반으로 inter-VLAN routing한다.

L3 스위치 SVI 예시

```
ip routing
!
interface Vlan10
 ip address 10.10.10.1 255.255.255.0
 no shutdown
!
interface Vlan20
 ip address 10.20.20.1 255.255.255.0
 no shutdown
```

Subinterface 번호와 VLAN ID

`G0/0.10`의 `.10`은 논리 인터페이스 이름의 번호일 뿐이다. 실제 VLAN 매핑은 `encapsulation dot1Q 10`이 정한다. 관례상 같은 번호를 써 가독성을 높이는 것이다.

EtherChannel

필기 확인 + Cisco/IEEE 보강

여러 흐름을 분산하지만 한 흐름은 보통 한 멤버에 고정된다

함께 찾을 말: Port-Channel, Link Aggregation, LAG, LACP, PAgP

EtherChannel란 여러 물리 Ethernet 링크를 하나의 논리 링크로 묶어 총 대역폭, 링크 장애 복원력, STP 단순화를 제공하는 링크 집성 기술이다.

EtherChannel이란

EtherChannel은 여러 물리 Ethernet 링크를 하나의 논리 포트 채널로 묶는 기술이다. STP에는 하나의 논리 링크처럼 보이므로 멤버 링크를 개별 중복 경로로 차단하지 않고, 링크 장애 시 남은 멤버로 통신을 유지할 수 있다. 표준 LACP와 Cisco PAgP/정적 on 방식이 있다.

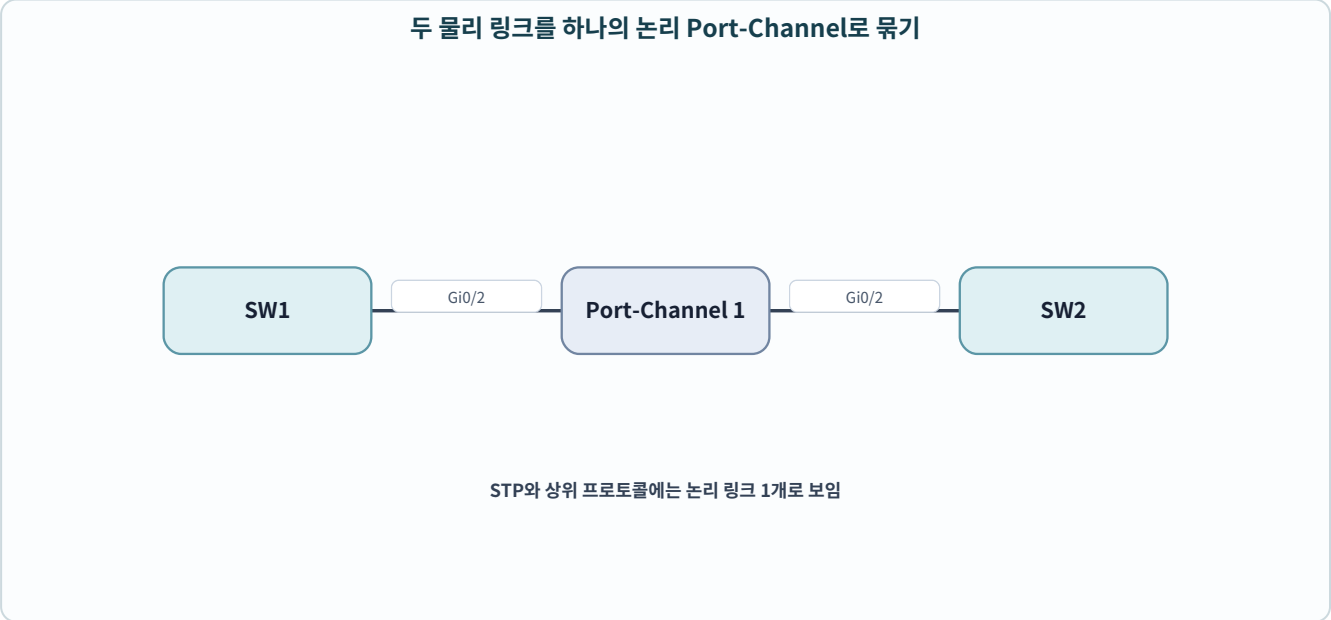


그림 9-6. 멤버 포트의 속도·duplex·access/trunk·allowed VLAN 같은 핵심 설정이 맞아야 한다.

왜 한 TCP 연결이 두 배 빨라지지 않을 수 있는가

패킷 순서가 뒤바뀌는 것을 줄이기 위해 스위치는 출발지/목적지 MAC, IP, L4 포트 등을 해시해 한 흐름을 특정 멤버 링크에 고정한다. 여러 흐름 전체는 분산되지만, 단일 5-tuple 흐름은 한 링크 용량에 제한될 수 있다.

잘못 구성되면 왜 루프·플래핑이 생기는가

한쪽은 Port-Channel로 묶었는데 반대쪽은 독립 포트로 인식하거나 VLAN 설정이 다르면, 같은 프레임이 예상 밖 경로로 들어와 MAC flapping과 STP 이상이 생길 수 있다. 양쪽 channel protocol, group number, trunk/access 설정을 대조해야 한다.

LACP EtherChannel 예시

```
interface range GigabitEthernet0/1-2
 switchport mode trunk
 channel-group 1 mode active
!
interface Port-channel1
 switchport mode trunk
 switchport trunk allowed vlan 10,20,30

show etherchannel summary
show lacp neighbor
```

Port Security

필기 확인 + Cisco IOS XE 검증

MAC 기반 제한이지 완전한 사용자 인증은 아니다

함께 찾을 말: 포트 시큐리티, switchport port-security

Port Security란 스위치 포트에서 허용할 출발지 MAC 주소와 개수를 정하고, 허용 범위를 벗어난 프레임에 위반 조치를 적용하는 L2 접속 제한 기능이다.

포트 시큐리티란 무엇인가

Port Security는 스위치의 특정 포트에서 허용할 출발지 MAC 주소 또는 최대 MAC 주소 수를 제한하는 기능이다. 벽면 포트에 허가된 프린터 하나만 있어야 하는데 다른 노트북이나 소형 스위치가 연결되는 상황을 줄이는 데 사용한다.

왜 필요한가

일반 스위치는 새 출발지 MAC을 자동 학습한다. 공격자가 빈 포트에 장치를 연결하거나 기존 장치를 뽑고 자기 장치를 꽂으면, 스위치는 그 MAC도 정상적으로 배운다. Port Security는 "이 포트에서 몇 개, 어떤 MAC까지 정상으로 인정할 것인가"라는 제한을 추가한다.

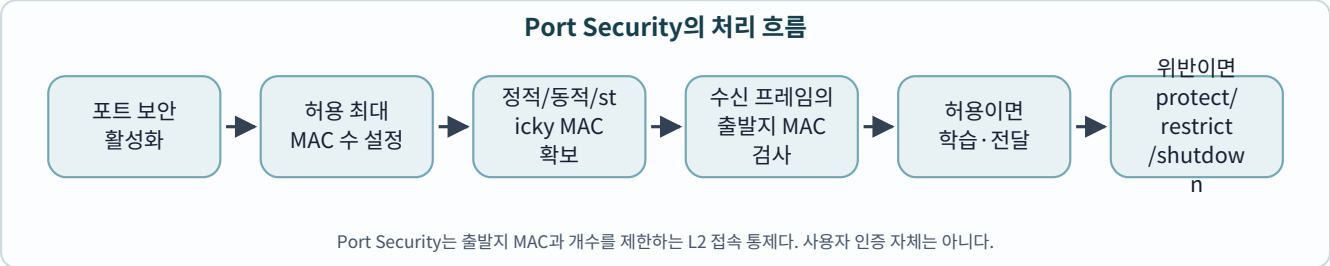


그림 9-7. 설정된 secure MAC 집합과 최대 개수를 기준으로 각 수신 프레임의 출발지 주소를 검사한다.

위반이 발생하는 대표 조건

- 이미 최대 secure MAC 수에 도달했는데 새로운 출발지 MAC이 들어온다.
- 한 secure port에 묶인 MAC이 같은 VLAN의 다른 secure port에서 보인다.
- 수동으로 지정한 secure MAC과 다른 장치가 프레임을 보낸다.

Port Security가 하지 못하는 것

- MAC 주소는 소프트웨어로 변경·스푸핑할 수 있으므로 강한 사용자 신원 인증이 아니다.
- 허용 MAC의 장치가 악성 코드에 감염된 것을 판별하지 못한다.
- ARP 스푸핑, DHCP 위조, IP 기반 공격을 모두 막지 못한다.
- 여러 사용자가 전화기 뒤에 연결되는 포트 등에서는 최대 MAC 수·voice VLAN 설계를 조정해야 한다.

802.1X와 Port Security를 구분하기

Port Security는 MAC 주소 학습 제한이고, IEEE 802.1X는 supplicant-authenticator-authentication server 구조로 네트워크 접속 전에 신원을 인증한다. 플랫폼과 설계에 따라 두 기능의 동시 사용 제약이 있으므로 제품 문서를 확인해야 한다.

Secure MAC과 Sticky 학습

필기 확인 + Cisco IOS XE 검증

자동 학습과 영구 저장은 같은 동작이 아니다

함께 찾을 말: Secure MAC Address, Static Secure MAC, Dynamic Secure MAC, Sticky Port, mac-address sticky

Secure MAC과 Sticky 학습란 Port Security가 허용 대상으로 관리하는 MAC 주소와, 포트가 동적으로 배운 주소를 running configuration에 붙여 넣어 고정하는 학습 방식이다.

종류	어떻게 생기는가	재부팅/설정 저장 시 성격
Static secure MAC	관리자가 MAC을 명시적으로 입력	running/startup configuration에 저장 가능
Dynamic secure MAC	포트가 자동 학습	일반적으로 주소 표에만 있고 재부팅 시 사라짐
Sticky secure MAC	자동 학습한 MAC을 running configuration에 sticky 항목으로 기록	copy running-config startup-config를 해야 재부팅 후 유지

Sticky의 말뜻과 실제 동작

Sticky는 “달라붙는다”는 뜻이다. 관리자가 수백 대 프린터의 MAC을 일일이 입력하지 않고, 최초 정상 연결 때 포트가 자동 학습한 MAC을 secure MAC으로 고정해 running-config에 기록한다. 하지만 저장하지 않으면 재부팅 후 사라질 수 있다.

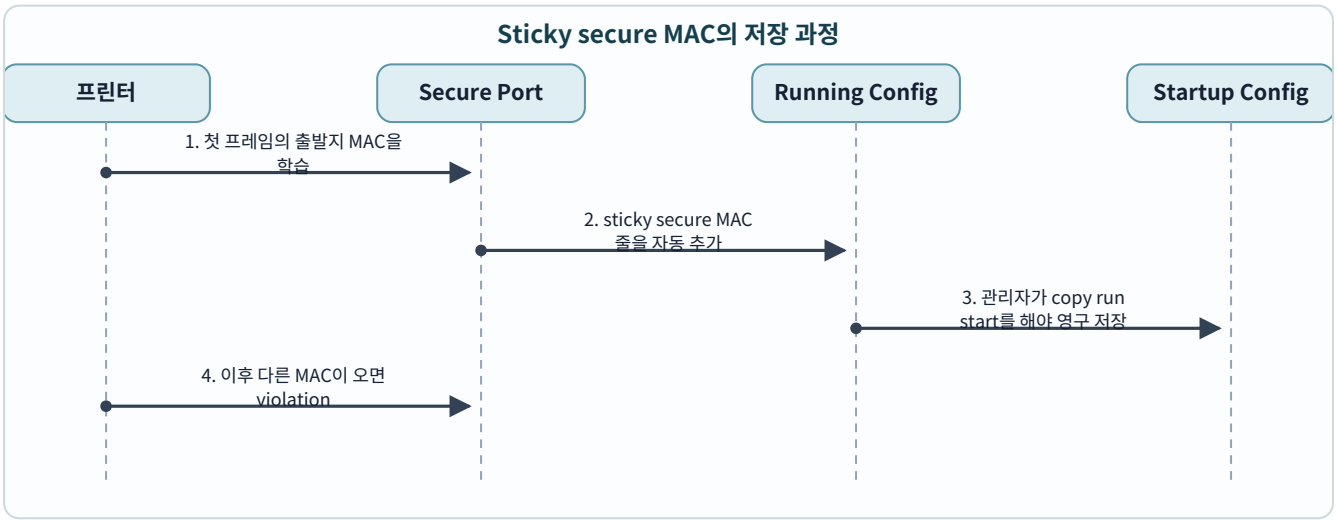


그림 9-8. Sticky는 자동 학습과 설정 기록을 결합한다. running-config와 startup-config는 자동으로 같아지지 않는다.

Sticky MAC 설정과 확인

```
interface GigabitEthernet0/10
 switchport mode access
 switchport access vlan 20
 switchport port-security
 switchport port-security maximum 1
 switchport port-security mac-address sticky
 switchport port-security violation restrict

show port-security interface gigabitEthernet0/10
show port-security address
show running-config interface gigabitEthernet0/10
copy running-config startup-config
```

처음 연결한 장치가 곧 "신뢰 장치"가 된다

설치 전에 공격 장치가 먼저 연결되어 sticky로 학습되면 오히려 그 MAC이 허용된다. 초기 학습 시 물리 보안, 변경 절차, 확인·저장이 필요하다.

Port Security violation mode

필기 확인 + Cisco IOS XE 검증

차단 강도와 가용성·가시성의 균형

함께 찾을 말: protect, restrict, shutdown, Violation Mode, error-disabled

Port Security violation mode란 허용되지 않은 MAC이 나타났을 때 프레임만 버릴지, 알림을 남길지, 포트 전체를 error-disabled로 내릴지 정하는 위반 처리 정책이다.

Violation mode	비허용 프레임	포트 상태	로그·카운터
protect	드롭	up 유지	일반적으로 알림 없음
restrict	드롭	up 유지	violation counter, syslog/SNMP 알림
shutdown	드롭 후 포트 error-disabled	down/err-disabled	로그와 카운터, 기본 모드인 경우가 많음
shutdown VLAN (일부 trunk 기능)	해당 VLAN만 error-disabled 처리 가능	플랫폼 의존	제품 문서 확인

Protect

허용되지 않은 출발지 MAC의 프레임만 조용히 버리고 포트는 계속 정상 MAC 트래픽을 전달한다. 운영자가 위반을 알아채기 어려워 문제 조사에 불리할 수 있다.

Restrict

비허용 프레임을 버리되 포트는 유지하고, 위반 카운터 증가와 syslog/SNMP 알림을 남긴다. 가용성을 유지하면서 탐지할 수 있지만 공격 프레임 처리는 계속 발생한다.

Shutdown

위반 시 포트를 error-disabled 상태로 만들어 모든 트래픽을 멈춘다. 강한 차단이지만 잘못된 최대 MAC 수, IP 전화기+PC, VM/컨테이너 등 정상 다중 MAC 환경에서 서비스 장애를 만들 수 있다. 복구 전 원인을 확인한다.

위반 상태 확인과 복구 개념

```
show interfaces status err-disabled
show port-security interface gi0/10

# 원인 제거 후 수동 복구 예시
interface gi0/10
 shutdown
 no shutdown

# 자동 복구는 정책과 위험을 검토
errdisable recovery cause psecure-violation
errdisable recovery interval 300
```

모드 선택은 보안 강도만의 문제가 아니다

무인 프린터 포트라면 shutdown이 적절할 수 있지만, 전화기 뒤 PC가 연결되거나 가상화 호스트처럼 여러 MAC이 정상인 포트에는 별도 설계가 필요하다. 포트 역할, 최대 MAC, aging, 음성 VLAN, 운영 대응 절차를 함께 정한다.

DHCP Snooping

필기 보강 + Cisco 검증

DHCP 메시지를 “엿보아” 신뢰 가능한 바인딩을 만든다

함께 찾을 말: DHCP 스누핑, Snooping Binding Database, Trusted Port, Untrusted Port

DHCP Snooping란 스위치가 DHCP 서버 방향을 신뢰 포트로 제한하고 정상 임대의 IP-MAC-VLAN-port 관계를 기록하여 가짜 DHCP 서버와 주소 위조를 줄이는 L2 보안 기능이다.

DHCP Snooping이란

DHCP Snooping은 스위치가 DHCP 메시지를 관찰해, 정식 서버 방향의 **trusted port**와 사용자 방향의 **untrusted port**를 구분하고, 비신뢰 포트에서 들어오는 가짜 서버 응답을 차단하며 정상 임대 정보를 바인딩 데이터베이스에 기록하는 기능이다.

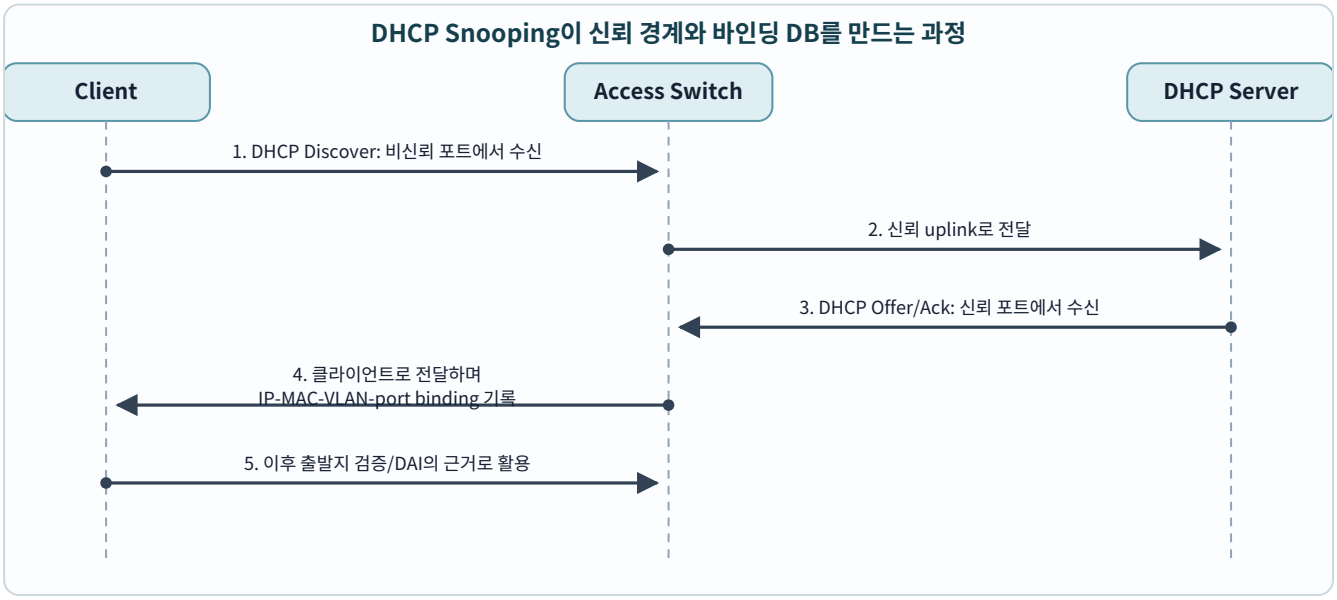


그림 9-9. 정상 서버 응답은 trusted uplink에서 와야 한다. 스위치는 ACK 등을 보고 IP-MAC-VLAN-port-lease 관계를 기록한다.

무엇을 방어하는가

공격자가 사용자 포트에서 DHCP Offer를 먼저 보내 자기 장치를 gateway/DNS로 지정하는 rogue DHCP 공격을 줄인다. 또한 바인딩 데이터는 Dynamic ARP Inspection, IP Source Guard의 검증 근거가 된다.

설계 시 주의

- trusted 포트를 잘못 지정하면 공격 서버 응답을 허용하거나 정상 DHCP를 모두 차단한다.
- 정적 IP 장치는 동적 DHCP binding이 없으므로 DAI/IPSG와 함께 쓸 때 별도 정책이 필요하다.
- Option 82 삽입과 DHCP relay 환경의 호환성을 확인한다.
- VLAN별 활성화 여부와 rate limit을 확인한다.

DHCP Snooping 개념 설정 예시

```
ip dhcp snooping
ip dhcp snooping vlan 10,20
!
interface GigabitEthernet0/1
 description UPLINK-TO-DHCP
 ip dhcp snooping trust
!
interface range GigabitEthernet0/2-24
 ip dhcp snooping limit rate 15

show ip dhcp snooping
show ip dhcp snooping binding
```

Dynamic ARP Inspection

필기 확인 + Cisco 검증

ARP의 "주장"을 검증 가능한 데이터와 대조한다

함께 찾을 말: DAI, 동적 ARP 검사, ARP Spoofing 방어

Dynamic ARP Inspection란 스위치가 비신뢰 포트의 ARP 메시지에 들어 있는 IP-MAC 관계를 DHCP Snooping 바인딩이나 ARP ACL과 대조해 위조 ARP를 차단하는 기능이다.

DAI가 필요한 이유

ARP는 기본적으로 인증 없이 "이 IP는 이 MAC이다"라는 주장을 받아들인다. 공격자가 gateway IP에 자기 MAC을 연결하는 가짜 ARP 응답을 보내면 피해자의 ARP cache가 오염될 수 있다. DAI는 스위치가 ARP 메시지의 주소 관계를 검증한다.

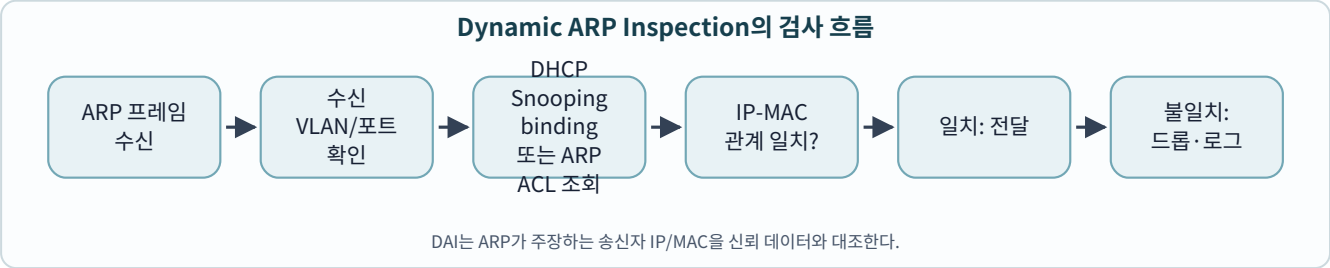


그림 9-10. DHCP Snooping 바인딩이 있는 동적 주소 환경에서는 ARP의 sender IP/MAC/port/VLAN 관계를 대조한다.

Trusted/Untrusted

일반 사용자 포트는 untrusted로 두어 ARP를 검사하고, 다른 스위치·라우터 방향 trusted port는 검사를 면제할 수 있다. 그러나 uplink를 무조건 신뢰하면 그 방향에서 들어오는 위조를 놓칠 수 있으므로 신뢰 경계를 실제 토폴로지에 맞게 설계한다.

정적 IP와 ARP ACL

서버·네트워크 장비가 정적 IP를 사용하면 DHCP binding이 없다. 이때 ARP ACL이나 정적 binding 같은 별도 검증 정보를 제공하지 않으면 정상 ARP가 드롭될 수 있다.

DAI 개념 설정 예시

```
ip arp inspection vlan 10,20
!
interface GigabitEthernet0/1
 ip arp inspection trust
!
show ip arp inspection
show ip arp inspection statistics
```

DAI는 암호화 기능이 아니다

DAI는 ARP 위조를 줄이는 L2 검증 기능이다. 통신 내용의 기밀성·무결성을 제공하지 않으므로 TLS/SSH 같은 종단 암호화가 별도로 필요하다.

이 장의 통합 실습

토폴로지: 스위치 2대, VLAN 10/20, trunk 1개, VLAN 10 PC 2대, VLAN 20 PC 1대, router-on-a-stick 또는 L3 SVI.

1. access/trunk를 구성하고 `show interfaces trunk`, `show vlan brief`로 확인한다.
2. Wireshark에서 trunk 캡처를 수행해 `vlan.id == 10`과 0x8100 TPID를 확인한다.
3. native VLAN을 한쪽만 바꿔 경고·통신 이상을 관찰한 뒤 즉시 복구한다.
4. VLAN 10 PC 포트에 Port Security maximum 1 + sticky + restrict를 설정한다.
5. 다른 MAC 장치를 연결해 violation counter와 로그, forwarding 상태를 비교한다.
6. DHCP Snooping/DAI를 켜기 전후 rogue DHCP/ARP 실습은 반드시 격리된 소유 실습망에서만 수행한다.

PART III

IP · 전송계층·라우팅·WAN의 메커니즘

한 LAN 안에서 프레임을 전달하는 법을 배웠다면, 이제 다른 네트워크까지 패킷을 보내는 판단과 종단 프로그램까지 데이터를 전달하는 과정을 살펴본다. 주소·테이블·타이머·메트릭이 실제 장비에서 어떤 순서로 사용되는지를 중심으로 설명한다.

10장. OSI 계층과 물리 통신: 신호에서 애플리케이션까지

> OSI 7계층은 현실 장비를 일곱 상자에 강제로 넣는 분류표가 아니라, 통신 문제를 서로 다른 책임으로 나누어 말하기 위한 모델이다. 패킷이 안 갈 때 "케이블 신호인가, MAC 전달인가, IP 경로인가, TCP 연결인가, HTTP 응답인가"를 차례대로 좁힐 수 있다.

OSI 7계층과 PDU

필기 확인 + 보충 해설

장애 지점을 단계적으로 좁히는 공통 언어

함께 찾을 말: OSI 7 Layer, PDU, Encapsulation, SI, NI, Service Interface, Network Interface

OSI 7계층과 PDU란 OSI 모델은 통신 기능을 물리 신호부터 응용 메시지까지 일곱 책임 계층으로 나눈 참조 모델이고, PDU는 각 계층이 header 등을 붙여 다루는 데이터 단위다.

계층	핵심 질문	대표 단위·예
7 Application	사용자 기능을 어떤 메시지 규칙으로 제공하는가	HTTP, DNS, SMTP
6 Presentation	데이터 표현·압축·암호 형식을 어떻게 맞추는가	TLS 표현, JSON, UTF-8
5 Session	대화 상태와 재개를 어떻게 관리하는가	세션 개념, RPC 대화
4 Transport	어느 프로세스에, 얼마나 신뢰성 있게 전달하는가	TCP segment, UDP datagram, port
3 Network	다른 네트워크까지 어느 경로로 보낼 것인가	IP packet, router
2 Data Link	같은 링크에서 어느 인터페이스로 전달하는가	Ethernet frame, switch, MAC
1 Physical	비트를 어떤 신호로 매체에 실을 것인가	전기·광·무선 신호, cable

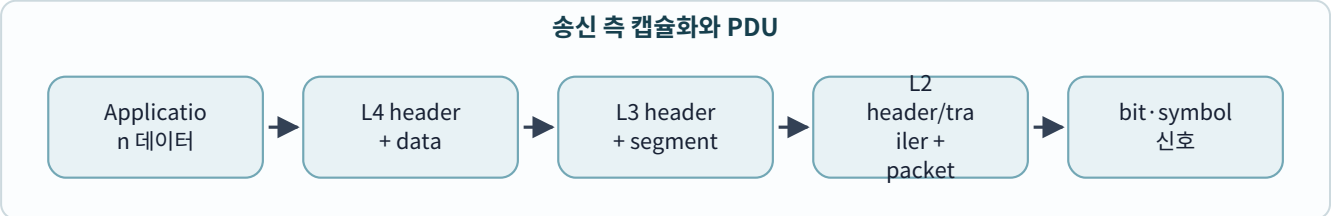


그림 10-1. 수신 측은 각 계층이 자신과 관련된 header를 검사하고 상위 계층으로 payload를 넘긴다.

PDU

PDU(Protocol Data Unit)는 특정 계층이 제어 정보를 붙여 취급하는 데이터 단위다. Ethernet에서는 frame, IP에서는 packet/datagram, TCP에서는 segment라는 이름을 쓴다. 실제 대화에서는 모두 “패킷”이라고 뭉뚱그리기도 하므로 정확한 계층을 같이 말하면 혼동이 줄어든다.

하드웨어와 소프트웨어 경계

L1은 PHY 회로와 매체가 중심이고, L2는 NIC·switch ASIC처럼 하드웨어 처리가 많다. 그러나 Linux bridge, virtual switch, software router처럼 같은 기능을 소프트웨어로 구현할 수 있다. 계층은 구현 재료가 아니라 기능 책임을 나눈다.

OSI와 TCP/IP 모델

Internet 실무에서는 link, internet, transport, application의 TCP/IP 모델도 널리 쓴다. OSI의 session·presentation 기능은 애플리케이션 라이브러리와 프로토콜 안에 섞여 구현되는 경우가 많다.

대역폭·지연·Duplex

필기 확인 + 보충 해설

빠른 회선이 항상 낮은 지연을 보장하지 않는 이유

함께 찾을 말: Bandwidth, Throughput, Latency, Simplex, Half Duplex, Full Duplex, Transceiver

대역폭·지연·Duplex란 대역폭은 링크의 이론적 전달 능력이고, 지연은 데이터가 도착하기까지 걸리는 시간이며, duplex는 두 방향 통신을 동시에 할 수 있는지를 나타낸다.

대역폭과 실제 처리량

대역폭(bandwidth)은 링크가 단위 시간에 운반할 수 있도록 설계된 비트율 또는 주파수 범위를 말한다. 1Gbps 링크라고 애플리케이션 데이터가 매초 정확히 1Gbit 전달되는 것은 아니다. Ethernet/IP/TCP header, inter-frame gap, 재전송, 혼잡, 디스크·CPU 성능을 뺀 실제 유효 처리량은 throughput이다.

지연을 구성하는 네 요소

Transmission delay

프레임의 모든 비트를 링크에 밀어 넣는 시간: frame size / link rate.

Propagation delay

신호가 매체를 물리적으로 이동하는 시간.

Processing delay

장비가 header를 검사하고 lookup하는 시간.

Queueing delay

출력 링크가 바빠 buffer에서 기다리는 시간.

단방향·반이중·전이중

simplex는 한 방향만, half-duplex는 양방향이지만 동시에 한 방향만, full-duplex는 양방향 동시 송수신이 가능하다. 현대 switched Ethernet의 point-to-point link는 보통 full-duplex라 CSMA/CD 충돌 검사를 사용하지 않는다. duplex mismatch는 한쪽이 full, 다른 쪽이 half로 인식해 late collision·오류·저성능을 만들 수 있다.

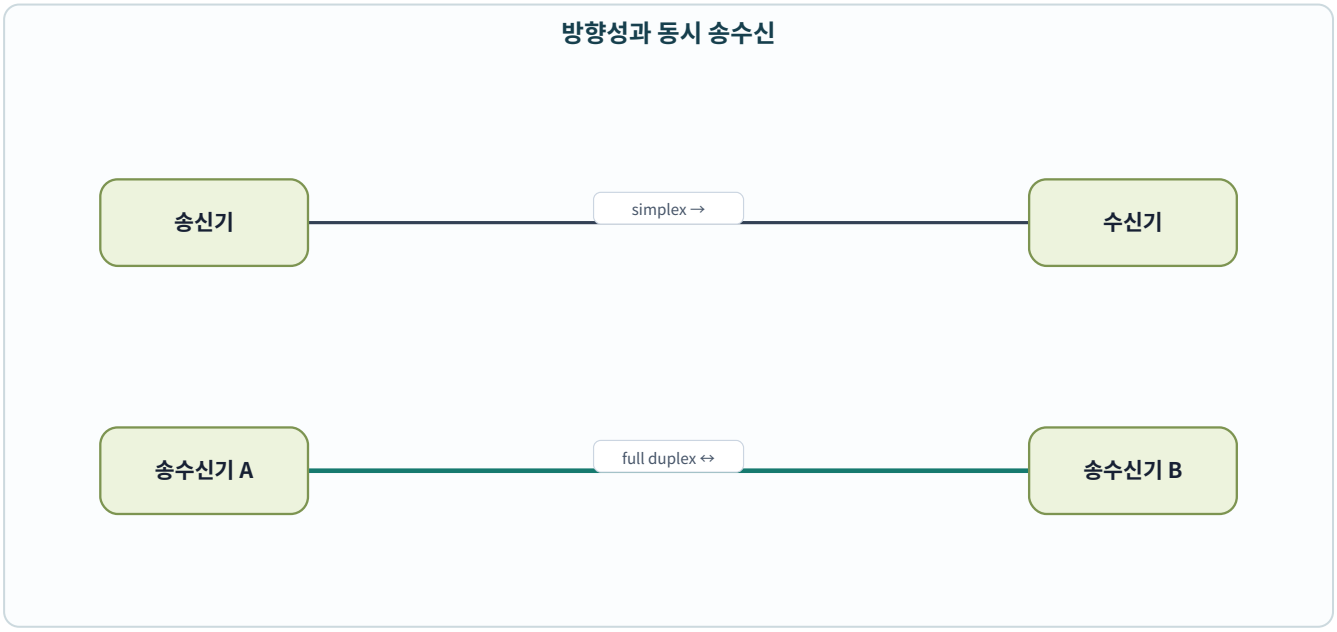


그림 10-2. transceiver는 송신기와 수신기 기능을 함께 가진 회로·모듈이다.

UTP·광섬유·배선 규격

필기 확인 + 보충 해설

비트를 실어 나르는 매체의 구조와 한계

함께 찾을 말: UTP, T568A, T568B, cross cable, coaxial, fiber, DAC, FDM, echo cancellation

UTP·광섬유·배선 규격란 UTP는 꼬임과 차동 신호로 잡음을 줄이는 구리 매체이고, 광섬유는 빛으로 정보를 전달하며, T568A/B는 twisted pair를 connector pin에 배치하는 표준이다.

UTP가 꼬여 있는 이유

twisted pair는 한 회로의 두 도체를 일정하게 꼬아 외부 전자기 잡음이 두 선에 비슷하게 유도되도록 한다. 수신기는 두 선의 전압 차를 읽는 differential signaling으로 공통 잡음을 상쇄한다. pair별 꼬임 간격을 다르게 해 pair 간 crosstalk도 줄인다.

T568A와 T568B

RJ-45로 흔히 부르는 8P8C connector에 네 pair를 배치하는 두 표준 pinout이다. 둘 다 성능상 동등하며 조직이 한 규격으로 일관되게 작업하는 것이 중요하다. 한쪽 A, 다른쪽 B면 과거 MDI/MDI-X 장비에서 crossover cable이 되었지만 현대 장비는 Auto-MDI/MDIX가 흔하다.

동축·광섬유·DAC

Coaxial cable

중앙 도체와 차폐층이 동심 구조인 케이블. 과거 Ethernet과 현재 RF·케이블망 등에 사용.

Optical fiber

빛의 반사·굴절을 이용해 전기적 간섭 없이 장거리·고속 전송. single-mode와 multi-mode가 있다.

DAC

Direct Attach Copper. 짧은 거리의 switch-server 연결에 쓰는 transceiver 일체형 twinax copper cable.

FDM과 echo cancellation

FDM은 서로 다른 주파수 대역을 나누어 여러 신호를 동시에 보내는 다중화 방식이다. echo cancellation은 송신 신호가 수신 경로로 되돌아오는 echo 성분을 추정해 빼는 기술이며 full-duplex copper 통신과 음성 시스템 등에서 중요하다.

Ethernet은 LAN 그 자체가 아니다

LAN은 제한된 지역의 네트워크 범주이고, Ethernet은 LAN을 구현하는 대표 기술 표준 집합이다. Wi-Fi도 LAN을 구성할 수 있고, Ethernet link가 WAN 서비스 일부로 쓰일 수도 있다.

오류 제어·ARQ·Hamming Code

필기 확인 + 보충 해설

모든 계층이 같은 방식으로 신뢰성을 제공하지 않는다

함께 찾을 말: ARQ, error control, Hamming code, CRC, FCS

오류 제어·ARQ·Hamming Code란 오류 제어는 전송 중 bit 손상을 검출·정정하거나 재전송하는 기술의 집합이고, ARQ는 응답과 timeout을 이용한 재전송, Hamming code는 여분 bit로 오류 위치를 판별하는 부호화 방식이다.

오류 검출과 오류 정정

오류 검출은 수신 데이터가 손상되었는지 알아내는 것이고, 오류 정정은 추가 정보로 원래 값을 복원하는 것이다. Ethernet FCS의 CRC는 검출 중심이며 잘못된 frame은 버리고 상위 계층이 재전송을 담당할 수 있다. Hamming code는 parity bit 위치를 설계해 일정 수의 bit 오류를 검출·정정한다.

ARQ

ARQ(Automatic Repeat reQuest)는 receiver가 ACK/NAK 또는 timeout으로 손실·오류를 알리고 sender가 다시 보내는 신뢰성 방식이다. Stop-and-Wait는 한 번에 하나, Go-Back-N은 오류 이후 window를 연속 재전송, Selective Repeat는 필요한 frame만 다시 보낸다. TCP의 sliding window와 selective acknowledgment는 이 계열의 아이디어를 사용하지만 세부 구현은 TCP 표준에 따른다.



그림 10-3. 오류 검출 정보와 재전송 제어를 조합해 신뢰성을 얻는다.

11장. IPv4 주소와 서브넷: 같은 네트워크를 판정하는 계산

> IP 주소는 장치의 절대적인 물리 위치가 아니라 특정 interface에 부여된 논리 주소다. host는 목적지 주소를 보고 “같은 L2 링크에서 직접 보낼지, gateway에 맡길지”를 subnet mask와 routing table로 판단한다.

IPv4 주소·Prefix·Network ID

필기 확인 + 보충 해설

주소 자체와 주소를 묶는 경계는 별도 정보다

함께 찾을 말: IP address, IPv4, Net ID, Host ID, Major Network, Class A, Class B, Class C

IPv4 주소·Prefix·Network ID란 IPv4 주소는 interface를 논리적으로 식별하는 32bit 값이고, prefix는 주소 앞부분 중 routing에서 같은 네트워크로 묶는 bit 수를 나타낸다.

주소는 interface에 붙는다

라우터에는 interface마다 다른 IP가 있고, 서버도 여러 NIC·VLAN subinterface·loopback에 여러 주소를 가질 수 있다. “컴퓨터의 IP”라는 말은 일상적 표현이며 실제 전달에서는 어느 interface와 prefix에 주소가 배정되었는지가 중요하다.

Network ID와 Host 부분

prefix length만큼의 앞 bit는 routing을 위한 network prefix, 나머지는 그 subnet 안의 interface를 구분하는 부분이다. 이는 번호 할당 규칙이지 packet 안에 network ID와 host ID가 별도 필드로 들어 있는 것은 아니다.

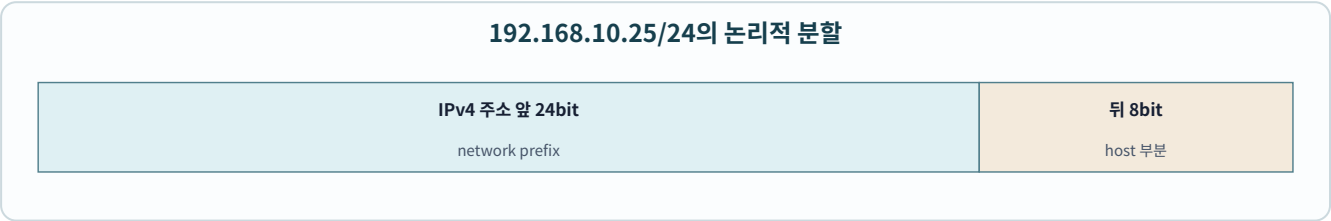


그림 11-1. mask 255.255.255.0과 /24는 앞 24bit가 prefix임을 나타낸다.

Major network와 classful 역사

과거 class A/B/C는 첫 bit 패턴으로 기본 network 경계를 정했다. major network는 classful 기본 경계를 가리키는 문맥에서 사용된다. 주소 낭비와 route table 문제 때문에 현재 Internet routing은 CIDR prefix를 사용하며 class 개념은 역사·일부 명령 동작 이해용으로 남아 있다.

IP가 다르면 통신이 되는 이유

IP 주소가 다르기 때문에 통신되는 것이 아니라, routing table에 목적지 prefix로 가는 경로가 있고 중간 router가 packet을 전달하기 때문에 통신된다. 같은 subnet의 서로 다른 IP는 직접 L2 전달하고, 다른 subnet이면 gateway를 사용한다.

Subnet Mask·CIDR·VLSM·Summary

필기 확인 + 보충 해설

같은 네트워크 판정과 주소 공간 설계의 수학

함께 찾을 말: Netmask, CIDR, Subnetting, VLSM, Supernetting, Route Aggregation, Route Summary

Subnet Mask·CIDR·VLSM·Summary란 subnet mask와 CIDR prefix length는 주소의 network bit 수를 나타내며, VLSM은 서로 다른 prefix 길이로 주소를 나누고, route summary는 공통 prefix를 하나로 묶어 광고한다.

Mask와 AND 연산

host는 자신의 IP와 목적지 IP에 subnet mask를 각각 bitwise AND해 network prefix를 비교할 수 있다. 결과가 같으면 on-link로, 다르면 routing table에서 next hop을 찾는다. 실제 OS는 connected route와 longest prefix match를 사용하지만 계산 원리는 같다.

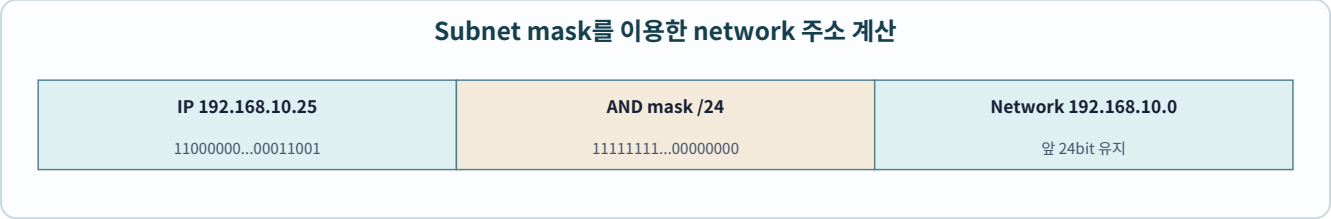


그림 11-2. mask의 1bit는 network 부분을 유지하고 0bit는 host 부분을 0으로 만든다.

/26 예제

/26은 host bit가 6개이므로 block size는 64다. 마지막 octet의 subnet 시작값은 0, 64, 128, 192다. 192.168.1.70/26은 64~127 범위에 있고 network address 192.168.1.64, broadcast 192.168.1.127, 일반 host 범위 65~126이다.

VLSM

VLSM은 하나의 상위 prefix를 필요한 host 수에 따라 서로 다른 길이의 subnet으로 나누는 방식이다. 큰 subnet부터 배치하면 주소 공간 겹침을 피하기 쉽다. route protocol이 prefix length를 함께 전달해야 VLSM을 지원한다.

Supernetting·Aggregation·Summary

여러 연속 prefix의 공통 앞 bit를 더 짧은 하나의 prefix로 광고해 route 수를 줄인다. summary는 관리·수렴에 유리하지만, 실제로 존재하지 않는 세부 subnet까지 포함하는 범위로 traffic을 끌어들일 수 있어 discard route와 설계 검증이 필요하다.

Subnet 계산 확인

```
# Linux가 계산한 주소 · route 확인
ip -4 addr show
ip route show
ip route get 192.168.1.70

# Python ipaddress 예시
python3 - <<'PY'
import ipaddress
n = ipaddress.ip_network('192.168.1.70/26', strict=False)
print(n.network_address, n.broadcast_address, n.num_addresses)
PY
```

사설 주소·공인 주소·IANA·ISP

필기 확인 + 보충 해설

주소 범위와 네트워크 신뢰도를 같은 말로 보지 않기

함께 찾을 말: Private IP, Public IP, RFC1918, IANA, ISP

사설 주소·공인 주소·IANA·ISP란 사설 IPv4 주소는 여러 조직이 내부에서 재사용하도록 global routing에서 제외된 주소 범위이고, 공인 주소는 Internet routing에 사용될 수 있으며, IANA와 registry·ISP가 주소 자원을 계층적으로 관리·제공한다.

사설 주소

IPv4 사설 대역은 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16이다. global Internet에서 직접 route하지 않도록 정해 조직 내부에서 재사용한다. 사설 주소라고 자동으로 안전한 것은 아니며 내부 접근 통제와 monitoring이 필요하다.

Public과 private network라는 말

public network는 Internet처럼 신뢰하지 않는 다수 주체가 접근할 수 있는 망을, private network는 조직이 통제하는 제한된 망을 뜻할 수 있다. 주소가 RFC1918인지와 보안 신뢰 수준은 같은 개념이 아니다. 공인 주소를 내부에서 쓰거나, 사설 주소 서비스가 VPN을 통해 외부에 노출될 수도 있다.

IANA와 ISP

IANA는 global IP address·AS number·protocol parameter의 최상위 조정을 수행하고 지역 Internet registry로 큰 주소 block을 배정한다. ISP는 고객에게 Internet 연결과 주소·route 서비스를 제공하는 사업자다.

범위	용도
10.0.0.0/8	사설
172.16.0.0/12	사설

범위	용도
192.168.0.0/16	사설
127.0.0.0/8	IPv4 loopback
169.254.0.0/16	link-local 자동 주소 문맥
224.0.0.0/4	IPv4 multicast

IPv6

필기 확인 + 보충 해설

주소 길이만 늘어난 것이 아닌 L3 동작 체계

함께 찾을 말: IPv6, Hop Limit, Neighbor Discovery, ICMPv6

IPv6란 IPv6는 128bit 주소와 개선된 header·자동 구성·neighbor discovery를 제공하는 Internet Protocol 버전이며, IPv4 주소 고갈과 확장성 문제를 완화하기 위해 설계되었다.

IPv6가 필요한 주된 이유

IPv4 32bit 주소의 부족과 NAT 의존을 완화하고 더 큰 주소 공간, 단순화된 기본 header, 자동 구성, multicast 중심 neighbor discovery 등을 제공한다. IPv6가 보안을 자동 보장하거나 NAT를 절대 쓰지 않는다는 뜻은 아니다.

표기와 압축

IPv6는 128bit를 16bit씩 8개 hexadecimal group으로 표기한다. 각 group의 앞 0을 생략할 수 있고, 연속된 0 group은 주소에서 한 번만 ::로 줄일 수 있다. prefix length는 IPv4와 같은 /n 형식이다.

Hop Limit과 Neighbor Discovery

IPv6의 Hop Limit은 IPv4 TTL과 같은 loop 제한 역할을 한다. ARP 대신 ICMPv6 Neighbor Discovery가 neighbor address resolution, router discovery, duplicate address detection을 제공한다. broadcast 대신 multicast를 사용한다.

IPv6 기본 header의 큰 구조



그림 11-3. 옵션은 extension header chain으로 연결한다.

12장. ARP·ICMP·DHCP·NAT: IP가 실제 링크에서 동작하도록 돕는 기능

> IP 주소만 알아서는 Ethernet frame을 만들 수 없다. host는 next hop의 MAC을 알아야 하고, 자신의 주소·gateway·DNS를 받아야 하며, 오류와 경로 상태를 전달할 제어 메시지도 필요하다. ARP, DHCP, ICMP, NAT는 이 빈틈을 각각 다른 층에서 메운다.

ARP

필기 확인 + 보충 해설

IP packet을 Ethernet frame에 실을 마지막 연결 고리

함께 찾을 말: Address Resolution Protocol, ARP cache, neighbor table, Gratuitous ARP, Proxy ARP

ARP란 ARP는 같은 IPv4 link에서 routing으로 선택된 next-hop IP 주소에 대응하는 link-layer MAC 주소를 알아내는 프로토콜이다.

ARP가 찾는 것은 최종 목적지 MAC이 아닐 수 있다

목적지 IP가 같은 subnet이면 그 host의 MAC을 찾는다. 다른 subnet이면 packet의 최종 destination IP는 그대로 두고, frame의 destination MAC에는 default gateway interface의 MAC을 넣는다. 즉 ARP는 routing 결정 후 선택된 **on-link next hop**의 MAC을 찾는다.

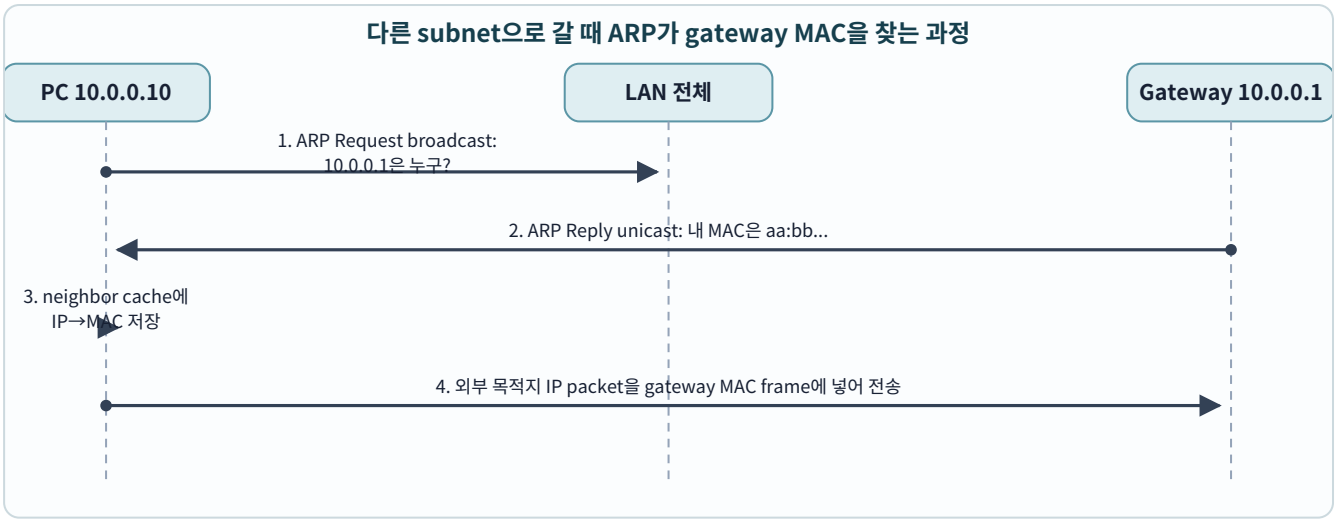


그림 12-1. IP packet의 destination은 원격 서버지만 Ethernet frame의 destination은 첫 router다.

ARP cache/neighbor table

응답을 매 packet마다 다시 묻지 않도록 IP→MAC mapping을 일정 시간 저장한다. Linux의 `ip neigh`는 REACHABLE, STALE, DELAY, PROBE 같은 neighbor 상태를 보여줄 수 있다. stale은 즉시 오류가 아니라 최근 확인되지 않았다는 뜻이다.

대표 ARP 변형

ARP Request/Reply

IPv4 next-hop address resolution의 기본 질의와 응답.

Gratuitous ARP

자기 IP를 대상으로 알리거나 확인하여 cache 갱신, duplicate 탐지, failover 알림에 사용.

ARP Probe

주소 사용 전 sender IP를 0.0.0.0으로 두고 충돌 여부를 확인하는 방식.

Proxy ARP

router가 다른 host를 대신해 자신의 MAC으로 ARP 응답하여 L2에 있는 것처럼 보이게 함.

Inverse/Reverse ARP 계열

특정 링크 기술에서 L2 식별자로 L3 주소를 알아내거나 초기 host address를 얻던 역사적 방식. DHCP가 일반적이다.

ARP table과 MAC table

ARP/neighbor table은 host·router가 IP→MAC을 저장한다. switch MAC table은 VLAN+MAC→port를 저장한다. ARP request가 switch를 지날 때 두 표가 동시에 갱신될 수 있어도 서로 다른 목적이다.

ICMP·Ping·Traceroute·TTL

필기 확인 + 보충 해설

"응답 없음"을 정확한 장애 지점으로 해석하기

함께 찾을 말: ICMP, ping, fping, traceroute, TTL, Time Exceeded

ICMP·Ping·Traceroute·TTL란 ICMP는 IP 전달 오류와 진단 정보를 전달하는 제어 프로토콜이고, ping은 Echo 메시지로 reachability와 RTT를 시험하며, traceroute는 TTL 만료 응답으로 hop을 추정한다.

ICMP는 단순 ping 프로토콜이 아니다

ICMP는 IP 전달 중 오류와 진단 정보를 전달한다. Echo Request/Reply가 ping에 쓰이고, Destination Unreachable, Time Exceeded, Redirect, Parameter Problem 등 다양한 message가 있다. application data transport용 TCP/UDP를 대신하지 않는다.

ping

ping은 ICMP Echo를 보내 왕복 시간과 손실을 관찰한다. ping 실패가 곧 host down을 뜻하지 않는다. firewall이 Echo를 막거나 return route가 없을 수 있다. 반대로 ping 성공은 특정 TCP port나 application이 정상이라는 보장이 아니다.

traceroute와 TTL

traceroute는 TTL/Hop Limit을 1부터 증가시킨 probe를 보낸다. 각 router에서 TTL이 0이 되면 ICMP Time Exceeded를 돌려보내므로 hop을 추정한다. 마지막 응답 방식은 UDP, ICMP, TCP probe 구현에 따라 다르다. router가 응답하지 않거나 load balancing을 하면 별표·경로 차이가 나타날 수 있다.

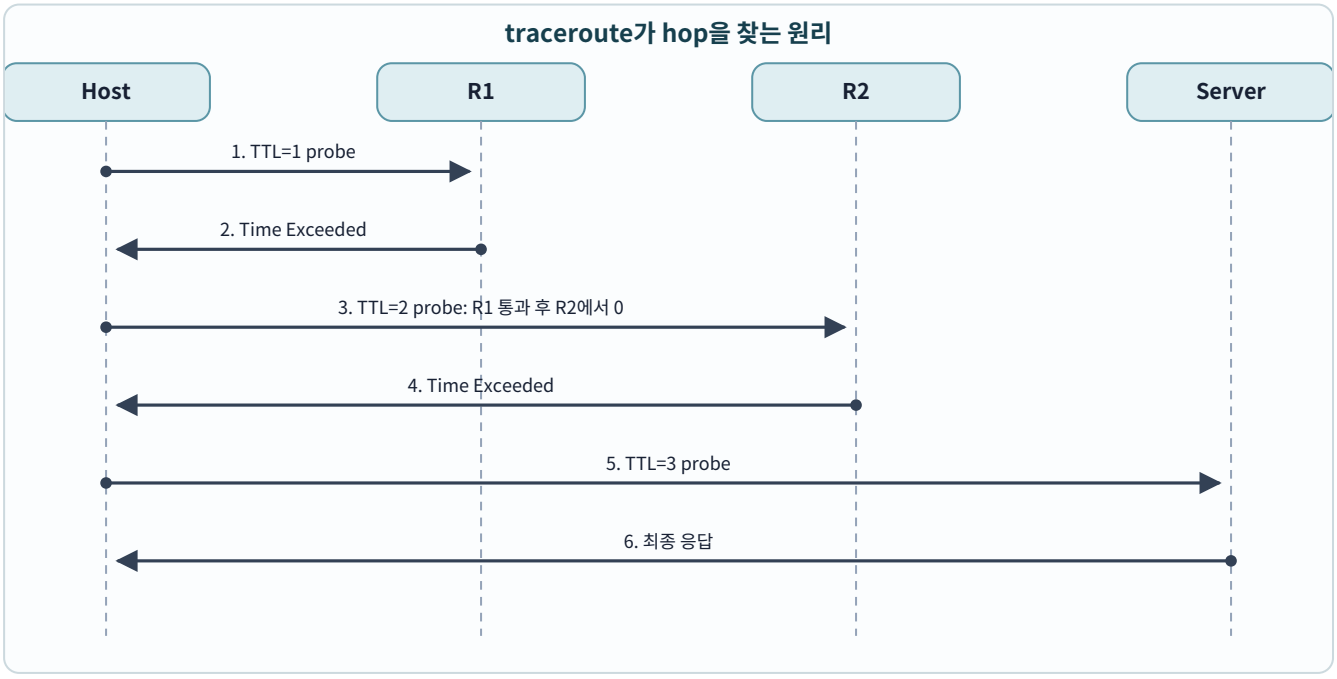


그림 12-2. TTL은 packet loop를 무한히 지속하지 않게 하는 L3 수명 값이다.

PMTUD와 fragmentation 관련 ICMP

Path MTU Discovery는 경로에서 허용되는 packet 크기를 찾는 데 ICMP 오류를 사용한다. 필요한 ICMP를 모두 막으면 큰 packet만 멈추는 black hole이 생길 수 있다. 보안 정책은 ICMP 전체 차단보다 message type과 필요성을 구분해야 한다.

DHCP

필기 확인 + 보충 해설

주소뿐 아니라 통신에 필요한 여러 option을 함께 배포한다

함께 찾을 말: Dynamic Host Configuration Protocol, DORA, DHCP Relay, nmcli

DHCP란 DHCP는 client에게 일정 기간 사용할 IP 주소와 subnet mask, gateway, DNS 등의 network 설정을 자동으로 임대하는 client-server 프로토콜이다.

DORA 네 단계

1. **Discover**: 주소가 없는 client가 server를 찾기 위해 broadcast 중심으로 요청한다.
2. **Offer**: server가 사용 가능한 IP와 lease·option을 제안한다.
3. **Request**: client가 선택한 offer를 알리고 해당 주소를 요청한다.
4. **Acknowledge**: server가 lease를 확정하고 설정을 전달한다.

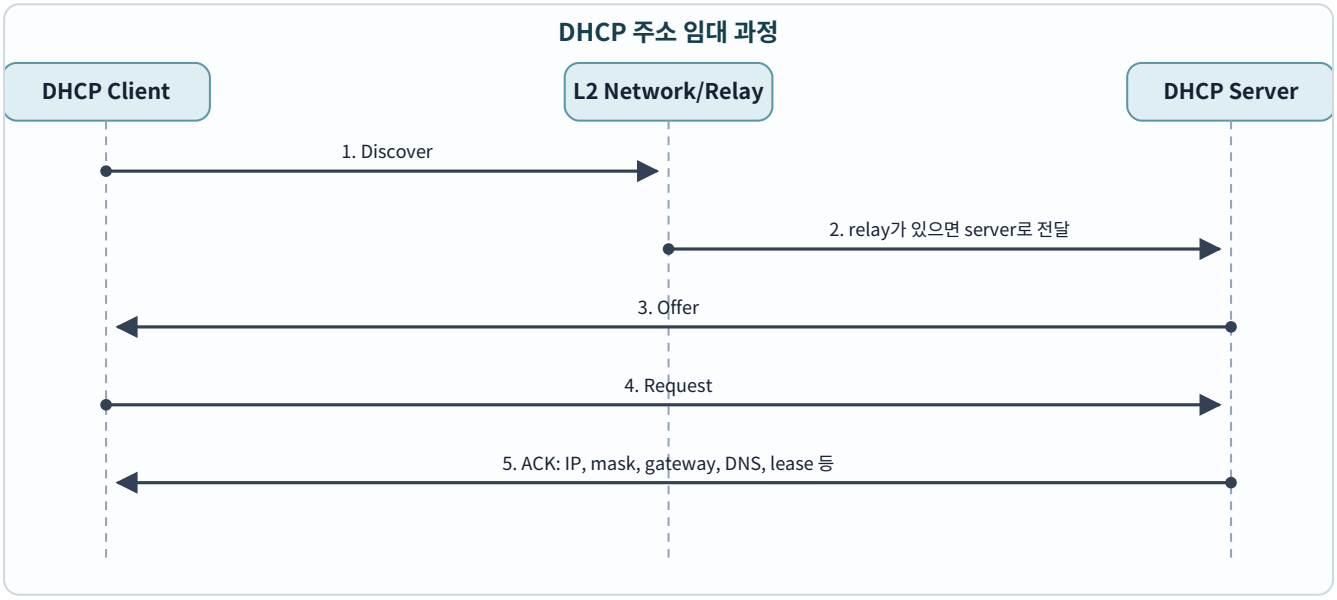


그림 12-3. 다른 subnet의 server를 쓰려면 router의 DHCP relay가 broadcast를 unicast로 중계할 수 있다.

DHCP가 전달할 수 있는 값

client IP, subnet mask, default gateway(router option), DNS server, domain search, lease time, NTP, PXE boot server/file 등 option을 전달할 수 있다. 모든 client가 모든 option을 사용하는 것은 아니다.

Server가 알아야 하는 값

주소 pool/range, subnet mask, gateway, DNS, lease 기간, 예약·제외 주소, 해당 subnet 식별이 필요하다. relay 환경에서는 giaddr 또는 relay option을 보고 어느 scope에서 주소를 줄지 판단한다.

DHCP로 받은 설정 확인

```
# Linux NetworkManager
nmcli device status
nmcli connection show
nmcli device show eth0

# lease/주소 확인
ip addr show
ip route show
resolvectl status
```

Rogue DHCP

공격자가 더 빠르게 응답해 자기 gateway·DNS를 배포하면 MITM이나 서비스 거부가 가능하다. switch의 DHCP Snooping은 신뢰 port에서 온 server reply만 허용하고 binding table을 만들 수 있다.

NAT·PAT·Masquerade·Floating IP

필기 확인 + 보충 해설

주소 부족 해결, network 경계, service 연속성을 서로 구분하기

함께 찾을 말: Network Address Translation, Static NAT, Dynamic NAT, PAT, NAPT, SNAT, DNAT, Masquerade, Floating IP

NAT·PAT·Masquerade·Floating IP란 NAT는 packet의 IP 주소와 경우에 따라 port를 변환하고 mapping state를 유지하는 기능이며, PAT는 port로 여러 flow를 구분하고, floating IP는 active 자원으로 이동 가능한 service 주소다.

NAT가 바꾸는 것

router/firewall이 packet의 source 또는 destination IP를 다른 주소로 바꾸고, transport port까지 바꾸는 경우가 있다. checksum도 변경 내용에 맞춰 갱신하고, return traffic을 원래 flow에 되돌리기 위한 translation/state table을 유지한다.

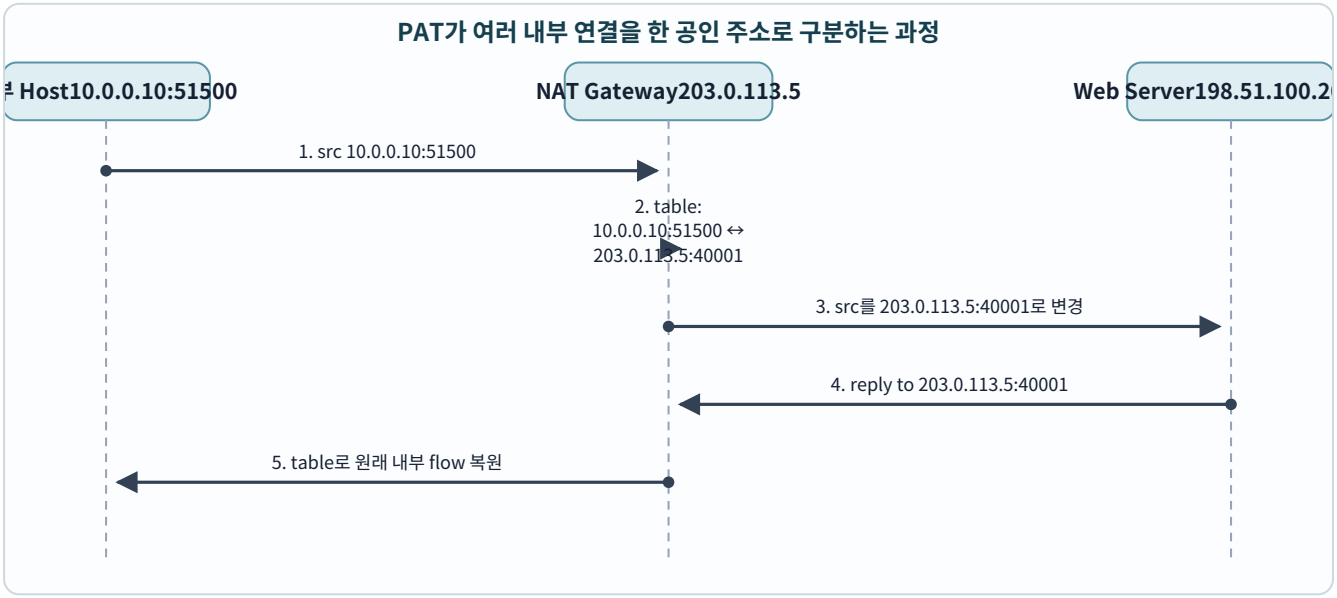


그림 12-4. NAT gateway의 state가 사라지면 진행 중 연결의 return mapping도 잃을 수 있다.

Static·Dynamic·PAT

Static NAT

내부 주소와 외부 주소를 고정 1:1 mapping. 서버 게시나 예측 가능한 mapping에 사용.

Dynamic NAT

공인 주소 pool에서 일시적으로 1:1 주소를 선택.

PAT/NAPT

여러 flow를 주소 하나 또는 소수 주소의 서로 다른 port로 구분. 가정용 Internet 공유에서 흔함.

DNAT/Port forwarding

들어오는 destination 주소·port를 내부 server로 변경.

Masquerade

Linux netfilter의 MASQUERADE는 outbound interface 주소가 동적으로 바뀌는 환경에서 그 interface의 현재 주소로 source NAT하는 target/동작을 뜻한다. 고정 주소에서는 명시적 SNAT이 더 예측 가능할 수 있다.

NAT와 방화벽은 다르다

NAT는 주소·port를 변환하고, firewall은 policy에 따라 traffic을 허용·차단한다. 많은 장비가 둘을 함께 제공해 NAT가 보안 기능처럼 느껴지지만, outbound mapping의 부수 효과와 명시적 접근 통제는 구분해야 한다.

Floating IP

floating은 특정 물리 장비에 영구 고정하지 않고 장애 조치나 orchestration에 따라 여러 node 중 현재 active node로 이동 가능한 논리 자원을 뜻한다. Floating IP는 client가 같은 service address를 계속 사용하게 하고, cloud에서는 NAT mapping이나 virtual router가 실제 port에 연결하는 방식으로 구현될 수 있다.

13장. Port·Socket·TCP·UDP와 주요 응용 프로토콜

> IP가 host interface까지 packet을 전달하면 운영체제는 어느 process가 받을지 결정해야 한다. Transport layer의 port와 socket이 이 역할을 하고, TCP는 순서·손실·흐름·혼잡을 관리하며 UDP는 최소한의 datagram 전달을 제공한다.

Port·Socket·5-tuple

필기 확인 + 보충 해설

IP가 장치를, port가 장치 안의 대화를 구분한다

함께 찾을 말: port, well-known port, socket, 5-tuple, listen, bind, accept

Port·Socket·5-tuple란 port는 host 내 transport endpoint를 구분하는 번호이고, socket은 application과 network stack 사이의 통신 endpoint이며, 5-tuple은 한 flow를 식별하는 주소·port·protocol 조합이다.

Port는 물리 구멍이 아니라 숫자 namespace다

transport port는 host 안의 service endpoint를 구분하는 16bit 번호다. TCP port 53과 UDP port 53은 서로 다른 protocol namespace라 동시에 다른 socket이 bind할 수 있다. “TCP와 UDP가 port를 공유한다”는 말은 같은 번호를 각각 쓸 수 있다는 뜻이지 같은 socket이라는 뜻이 아니다.

Well-known port

0~1023 범위는 전통적으로 well-known/system port로 관리된다. Linux에서 낮은 port bind는 privilege/capability가 필요할 수 있다. service가 표준 port를 쓴다는 것은 관례와 discovery 편의를 제공하지만 port 번호만으로 실제 application을 확정할 수 없다.

Socket과 5-tuple

socket은 application이 network stack을 사용하는 endpoint 객체다. 연결된 TCP flow는 source IP, source port, destination IP, destination port, transport protocol의 5-tuple로 구분한다. server는 한 listen port에서 accept한 여러 client connection을 각기 다른 5-tuple socket으로 관리한다.

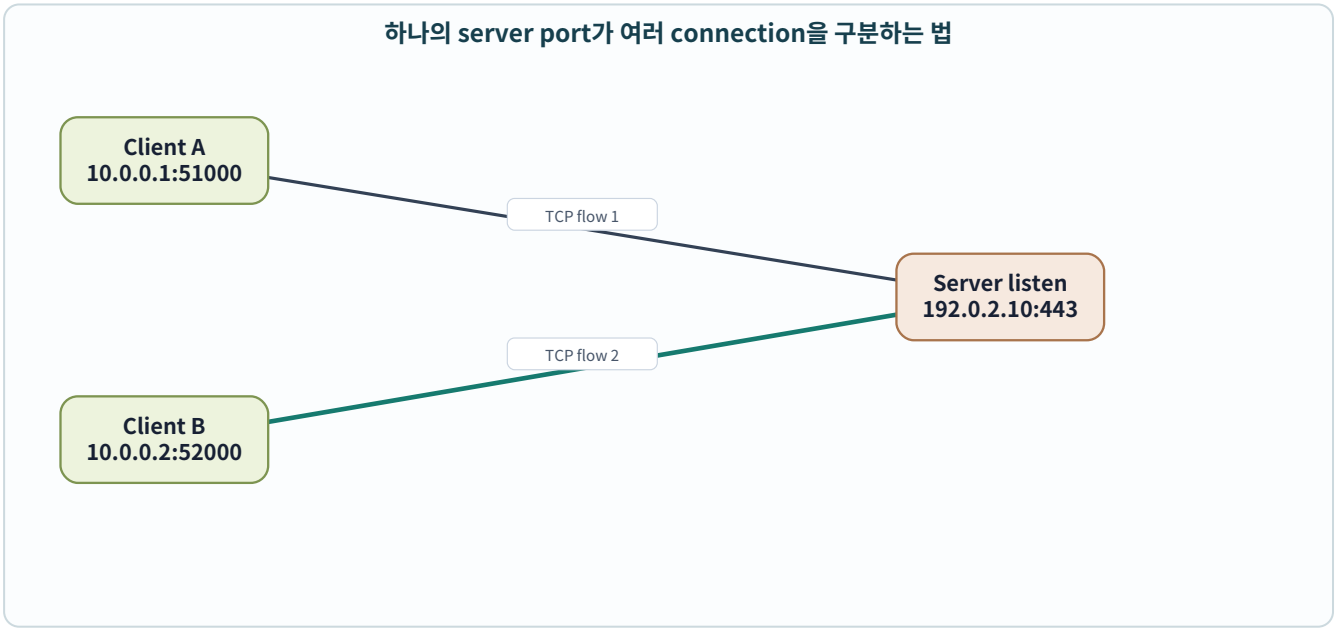


그림 13-1. destination port는 같아도 source address·port가 달라 5-tuple이 구분된다.

port와 socket 확인

```
ss -lntup
ss -tnp state established
lsof -iTCP:443 -sTCP:LISTEN
cat /etc/services | head
```

TCP 연결 설정과 종료

필기 확인 + 보충 해설

연결은 선이 아니라 양 끝의 상태 기계다

함께 찾을 말: TCP, 3-way handshake, 4-way handshake, SYN, ACK, FIN, RST, TIME_WAIT

TCP 연결 설정과 종료란 TCP는 SYN·ACK로 양쪽의 초기 sequence 번호와 연결 가능 상태를 확인하고, FIN으로 각 방향 stream을 질서 있게 닫거나 RST로 즉시 중단한다.

3-way handshake가 확인하는 것

1. Client가 SYN과 초기 sequence number(ISN)를 보낸다.
2. Server가 SYN+ACK로 자기 ISN을 보내고 client ISN+1을 확인한다.
3. Client가 server ISN+1을 ACK한다. 양쪽이 송수신 경로와 sequence space를 확인하고 ESTABLISHED가 된다.

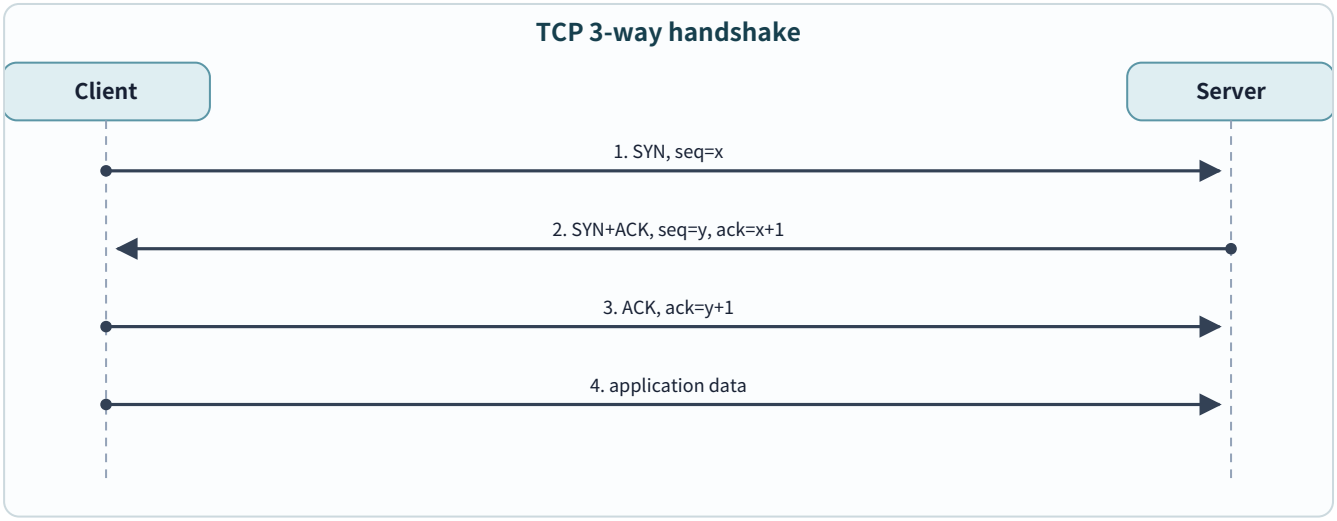


그림 13-2. ACK 번호는 “다음에 받고 싶은 byte sequence”를 뜻한다.

Sequence와 ACK

TCP는 application byte stream에 번호를 붙인다. segment가 1000번부터 500byte를 담으면 다음 기대 번호는 1500이다. ACK는 frame 개수가 아니라 연속해서 받은 byte 다음 위치를 누적 확인한다. 손실이 있으면 duplicate ACK와 retransmission timer가 재전송 단서가 된다.

4-way close와 FIN/RST

각 방향 byte stream을 독립적으로 닫기 때문에 한쪽 FIN, 상대 ACK, 상대 FIN, 마지막 ACK의 네 message가 흔하다. 실제 packet은 ACK와 FIN이 결합되어 줄 수 있다. RST는 정상적인 순차 종료 없이 connection을 즉시 중단하거나 존재하지 않는 socket을 알린다.

TIME_WAIT

마지막 ACK를 보낸 쪽이 일정 시간 TIME_WAIT에 머물러 늦게 도착한 이전 segment가 새 connection에 섞이는 것을 막고, 마지막 ACK가 손실되면 다시 보낼 수 있게 한다. 무조건 “불필요한 연결 찌꺼기”가 아니다.

SYN·ACK·FIN은 1byte 데이터인가

SYN과 FIN은 sequence space를 각각 1 소비하지만 application payload byte와 같은 내용은 아니다. TCP 상태 기계가 connection 시작·종료를 추적하도록 설계된 control flag다.

TCP 신뢰성·Sliding Window·흐름/혼잡 제어

필기 확인 + 보충 해설

“신뢰성”이 여러 독립 메커니즘의 결합인 이유

함께 찾을 말: Sequence Number, Acknowledgment Number, Sliding Window, Flow Control, Congestion Control, Retransmission, SACK

TCP 신뢰성·Sliding Window·흐름/혼잡 제어란 TCP는 sequence·ACK·재전송으로 byte stream의 순서와 손실을 관리하고, sliding window로 여러 데이터를 연속 전송하며, flow control과 congestion control로 receiver와 network의 처리 한계를 조절한다.

Sliding Window

sender가 ACK 하나를 기다린 뒤 다음 byte를 보내면 긴 RTT에서 link를 놀리게 된다. sliding window는 ACK를 기다리는 동안 일정 범위의 미확인 데이터를 연속 전송하고 ACK가 오면 허용 범위를 앞으로 미는 방식이다.

흐름 제어와 혼잡 제어

flow control은 receiver buffer가 넘치지 않도록 receiver advertised window(rwnd)를 사용한다. congestion control은 network path가 감당할 수 있는 양을 추정해 congestion window(cwnd)를 조정한다. 실제 송신 가능량은 둘 중 더 작은 제약과 기타 조건에 영향을 받는다.

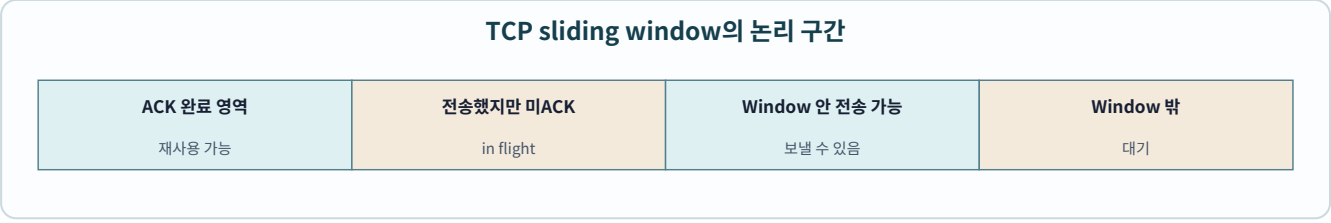


그림 13-3. ACK가 진행되면 창의 왼쪽 경계가 앞으로 이동한다.

혼잡이 발생하면

router queue가 쌓이고 drop 또는 ECN 표시가 발생할 수 있다. TCP는 ACK 패턴·timeout·ECN으로 혼잡을 추정해 전송률을 낮추고, 경로가 여유로우면 다시 늘린다. algorithm(CUBIC, BBR 등)에 따라 정책이 다르다.

SACK

누적 ACK만으로는 중간 손실 뒤에 무엇을 받았는지 정확히 알기 어렵다. Selective Acknowledgment option은 receiver가 이미 받은 비연속 byte block을 알려 sender가 필요한 구간 중심으로 재전송하도록 돕는다.

UDP·QUIC·SCTP

필기 확인 + 보충 해설

전송계층이 TCP와 UDP 두 개뿐은 아니다

함께 찾을 말: UDP, User Datagram Protocol, QUIC, SCTP

> **UDP·QUIC·SCTP란** UDP는 최소한의 port 기반 datagram 전달을 제공하고, QUIC은 UDP 위에서 보안·신뢰성·다중 stream을 구현하며, SCTP는 message와 multi-homing 기능을 제공하는 transport protocol이다.

UDP의 최소 기능

UDP는 source/destination port, length, checksum을 가진 datagram을 전달하지만 connection handshake, 순서 보장, 재전송, congestion control을 자체 제공하지 않는다. application이 필요한 신뢰성과 시간 제한을 직접 설계한다.

UDP를 쓰는 이유

작은 request/response, 실시간 음성·영상, multicast, application이 자체 transport를 구현하는 경우처럼 낮은 overhead와 메시지 경계 보존이 유리할 수 있다. “빠르다”는 것은 항상 지연이 낮다는 보장이 아니라 TCP의 기능을 기본으로 하지 않는다는 뜻이다.

QUIC과 SCTP

QUIC은 UDP 위에서 TLS와 신뢰성·multiplexed stream·congestion control을 구현하는 transport로 HTTP/3에 사용된다. SCTP는 message-oriented 전송, multi-streaming, multi-homing 같은 기능을 제공한다. 따라서 L4 protocol이 TCP/UDP 둘뿐이라는 설명은 불완전하다.

특성	TCP	UDP	QUIC
연결/상태	connection-oriented	기본적으로 없음	encrypted connection
데이터 모델	byte stream	datagram	여러 stream
신뢰성	기본 제공	application 선택	기본 제공
handshake	TCP + 별도 TLS	없음	transport+TLS 통합

Client-Server·P2P·Keepalive·Load Balancing

필기 확인 + 보충 해설

프로토콜 위에서 service를 지속 제공하는 구조

함께 찾을 말: Client, Server, P2P, Keep Alive, Round Robin, Load Balancer

Client-Server·P2P·Keepalive·Load Balancing란 client-server는 요청자와 제공자의 역할 분리 구조이고, P2P는 peer가 역할을 공유하며, keepalive는 비활성 구간의 생존 상태를 확인하고, load balancing은 여러 backend로 작업을 분배한다.

Client와 server는 역할이다

client는 service를 요청하고, server는 정해진 endpoint에서 요청을 기다려 응답한다. 한 프로그램이 다른 상황에서는 둘 다 될 수 있다. DNS resolver는 application에게 server지만 upstream DNS에는 client다.

P2P

peer-to-peer는 참여 node가 중앙 server에만 의존하지 않고 서로 client와 server 역할을 나누는 구조다. discovery, NAT traversal, 신뢰·무결성, version 관리가 별도 과제다.

Keepalive

오랫동안 application data가 없을 때 상대와 경로가 살아 있는지 확인하는 주기적 message다. TCP keepalive, HTTP persistent connection, routing protocol hello처럼 계층과 목적이 다르다. 너무 짧으면 불필요한 traffic과 battery/CPU 부담, 너무 길면 장애 탐지가 늦다.

Load balancing와 Round Robin

load balancer는 여러 backend에 request/connection을 분배한다. round robin은 순서대로, weighted round robin은 가중치만큼, least connections는 현재 연결 수가 적은 backend를, hash 방식은 client·URL 값을 기준으로 일관되게 선택할 수 있다. health check와 session persistence가 실제 가용성에 중요하다.

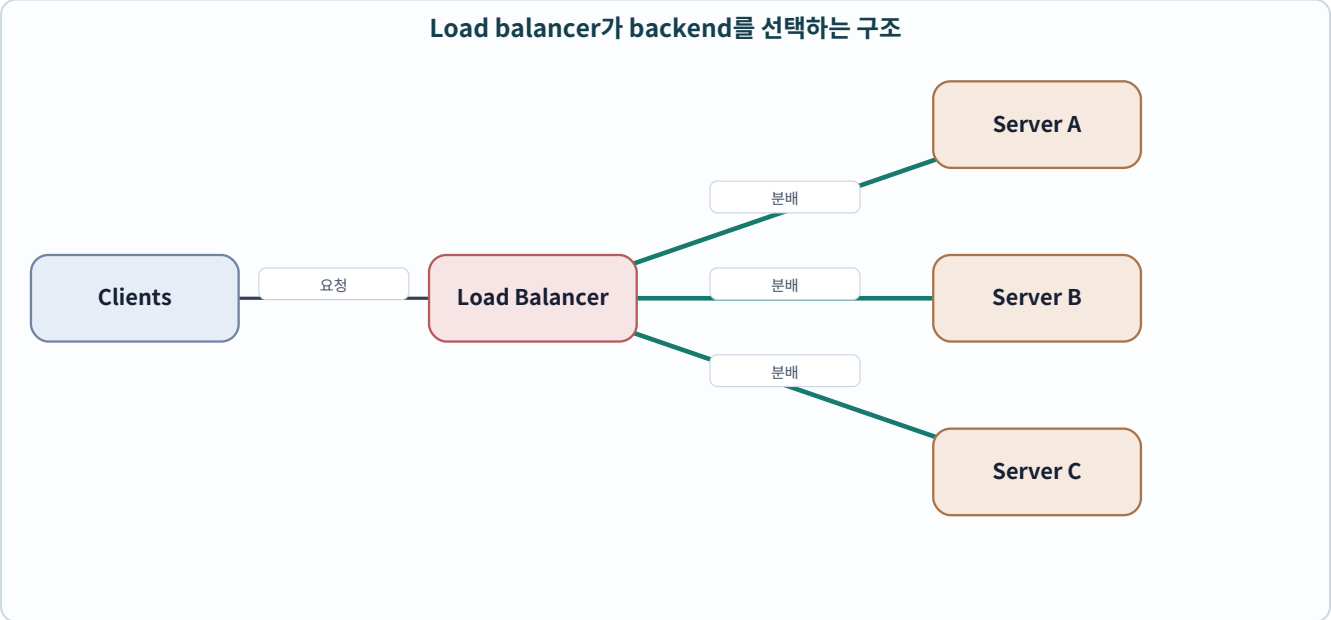


그림 13-4. 분배 알고리즘만으로 장애를 피할 수 없고 backend health를 지속 확인해야 한다.

주요 응용 프로토콜과 Well-known Port

필기 확인 + 보충 해설

번호 암기보다 요청·응답과 보안 특성을 연결하기

함께 찾을 말: HTTP, HTTPS, DNS, SSH, Telnet, FTP, SMTP, SNMP, NTP, MySQL, well-known port

주요 응용 프로토콜과 Well-known Port란 응용 프로토콜은 application message의 문법과 의미를 정하고, well-known port는 흔한 service endpoint를 발견하기 위한 등록·관례 번호다.

Protocol	기본 역할	자주 보이는 port·transport
HTTP/HTTPS	웹 request/response	TCP 80/443, HTTP/3는 QUIC/UDP 443
DNS	이름과 record 조회	UDP/TCP 53
SSH	암호화 원격 shell·tunnel	TCP 22
Telnet	평문 원격 terminal	TCP 23
FTP	제어와 별도 data connection	TCP 21 중심
SMTP	메일 server 간/제출	TCP 25, 587 등
SNMP	network 관리 조회·trap	UDP 161/162
NTP	시간 동기화	UDP 123
MySQL protocol	DB client-server	TCP 3306 관례

Port 번호는 application 신원증이 아니다

공격자는 443에서 SSH를 실행할 수 있고, application은 비표준 port를 쓸 수 있다. service 식별은 handshake, protocol parser, certificate, process 정보와 함께 판단해야 한다.

평문과 암호화

Telnet, FTP, HTTP처럼 기본 전송이 평문인 protocol은 같은 경로의 관찰자가 자격증명과 데이터를 읽거나 변경할 수 있다. SSH, TLS, IPsec 같은 암호화 channel은 기밀성뿐 아니라 무결성과 상대 인증을 함께 고려한다.

> **“SSL”이라는 말**
>
> SSL 2.0/3.0은 오래전에 폐기된 protocol이고 현대 보안 통신은 TLS를 사용한다. 제품 UI가 관습적으로 SSL 인증서라고 부르더라도 실제 협상 protocol과 허용 version을 확인해야 한다.

14장. 라우팅 표와 경로 선택: Router가 packet마다 하는 판단

> Router는 목적지 주소 전체와 정확히 같은 항목만 찾지 않는다. 여러 prefix 중 목적지와 가장 길게 일치하는 route를 고르고, 같은 prefix를 여러 source가 제안하면 administrative distance와 metric을 비교한다. 그 뒤 next hop의 L2 주소를 알아내 새 frame을 만든다.

Routing Table · Longest Prefix Match · Gateway

필기 확인 + 보충 해설

IP 주소 한 개를 보고 실제 출력 interface를 정하는 과정

함께 찾을 말: Routing Table, Longest Prefix Match, Next Hop, Gateway of Last Resort, Default Gateway, Adjacent, Adjacency

Routing Table · Longest Prefix Match · Gateway란 routing table은 destination prefix별 next hop과 정책 정보를 저장하고, router는 목적지와 가장 긴 prefix가 일치하는 route를 선택해 packet을 전달한다.

Routing table 항목의 구성

대표적으로 destination prefix, next hop 또는 outgoing interface, route source/protocol, metric, preference/administrative distance, age가 있다. connected route는 interface에 주소가 올라오면 생기고, local route는 장비 자신의 주소를 나타낼 수 있다.

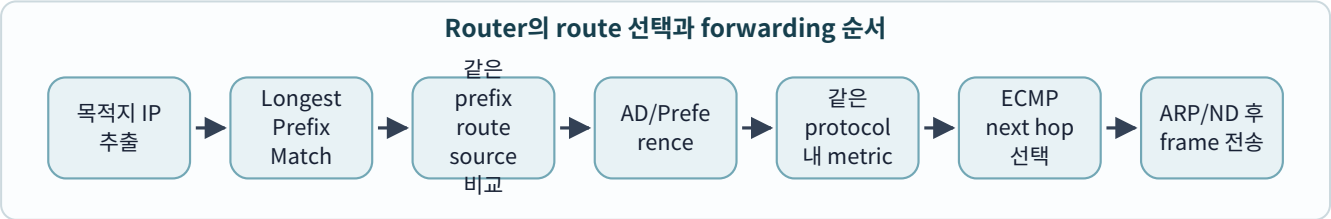


그림 14-1. 구현과 vendor에 따라 tie-break 세부는 다르지만 prefix specificity가 먼저다.

Longest Prefix Match

0.0.0.0/0, 10.0.0.0/8, 10.1.0.0/16이 모두 10.1.2.3과 일치하면 /16이 가장 구체적이므로 선택된다. AD가 더 낮은 default route가 있어도 더 긴 prefix route가 우선이다.

Next hop와 outgoing interface

point-to-point link에서는 interface만으로 충분할 수 있고, Ethernet 같은 multi-access에서는 next-hop IP와 그 MAC이 필요하다. recursive lookup은 route의 next-hop 주소로 가는 route를 다시 찾는 과정이다.

Gateway of Last Resort

더 구체적인 route가 없을 때 사용하는 default route를 가리키는 Cisco 표현이다. IPv4는 0.0.0.0/0, IPv6는 ::/0이다. default gateway는 host가 off-link 목적지를 맡길 router next hop이고, gateway라는 일반 용어는 protocol·system 사이 변환 지점을 더 넓게 뜻할 수 있다.

Routing과 forwarding 정보 확인

```
# Linux
ip route show
ip route get 8.8.8.8

# Cisco 계열
show ip route
show ip route 10.1.2.3
show ip cef 10.1.2.3 detail
```

Static · Dynamic Routing · Metric · AD · ECMP

필기 확인 + 보충 해설

"어느 경로가 더 좋은가"를 비교하는 서로 다른 단계

함께 찾을 말: Static Route, Dynamic Route, Default Route, Metric, Administrative Distance, AD, Floating Static, ECMP

Static · Dynamic Routing · Metric · AD · ECMP란 static route는 관리자가 지정하고 dynamic routing은 protocol로 학습하며, metric은 같은 protocol의 경로 비용, administrative distance는 서로 다른 route source의 신뢰 우선순위, ECMP는 같은 비용의 여러 next hop을 사용하는 방식이다.

Static · Default · Dynamic

Static route

관리자가 destination과 next hop을 직접 지정. 예측 가능하지만 변화에 자동 대응하지 않음.

Default route

더 구체적 route가 없을 때 사용하는 catch-all route.

Dynamic routing

router가 protocol message를 교환해 topology와 reachability 변화에 따라 route를 학습·철회.

Metric

한 routing protocol 안에서 후보 경로의 비용을 비교하는 값이다. RIP은 hop count, OSPF는 cost, EIGRP는 bandwidth와 delay 기반 composite metric을 기본 사용한다. 서로 다른 protocol의 metric 숫자는 단위와 의미가 달라 직접 비교하지 않는다.

Administrative Distance

같은 destination prefix를 서로 다른 route source가 제공할 때 어느 source를 더 신뢰할지 나타내는 local preference 값이다. Cisco IOS의 AD가 대표적이다. 낮을수록 우선한다. 같은 source/protocol 안의 경로는 metric을 비교한다. vendor별 명칭과 기본값이 다르다.

Floating Static Route

주 경로보다 높은 AD를 가진 static route를 미리 넣어 평소에는 routing table에 선택되지 않다가 주 동적 route가 사라지면 설치되도록 한 backup route다. 단순 interface up/down만으로 upstream end-to-end 장애를 감지하지 못할 수 있어 tracking과 함께 설계한다.

ECMP

Equal-Cost Multi-Path는 같은 prefix에 같은 비용의 여러 next hop을 설치해 load sharing과 redundancy를 제공한다. packet마다 무작위가 아니라 보통 5-tuple hash 등으로 flow를 일정 경로에 고정해 순서 뒤바뀜을 줄인다.

Hop · Convergence · Routing Loop · Stub Network

필기 확인 + 보충 해설

경로가 존재하는 것과 안정적으로 합의된 것은 다르다

함께 찾을 말: Hop, Convergence, Routing Loop, Stub Network, Blackhole

> **Hop · Convergence · Routing Loop · Stub Network란** hop은 L3 전달 한 단계, convergence는 변화 후 route 상태가 안정되는 과정, routing loop는 잘못된 next-hop 관계로 packet이 순환하는 상태이며, stub network는 transit 역할이 제한된 edge network다.

Hop

hop은 packet 또는 routing 정보가 한 L3 장비에서 다음 L3 장비로 이동하는 한 단계다. switch 통과는 일반적인 IP hop count에 포함되지 않는다. RIP hop count와 IP TTL은 모두 hop이라는 말을 쓰지만 하나는 route metric, 하나는 packet 수명이다.

Convergence

topology 변화가 발생한 뒤 모든 관련 router가 일관된 새 route를 계산하고 forwarding이 안정될 때까지의 과정과 시간을 말한다. detection, 정보 전파, route 계산, FIB 설치가 포함된다. 빠른 hello만으로 전체 convergence가 자동으로 빠른 것은 아니다.

Routing loop

router들이 서로를 목적지로 향하는 next hop으로 잘못 선택해 packet이 원형으로 도는 상태다. IP TTL이 결국 packet을 폐기하지만 그동안 대역폭과 CPU를 낭비하고 통신은 실패한다. dynamic protocol은 sequence, split horizon, topology database, feasibility condition 등 각기 다른 방식으로 loop를 방지한다.

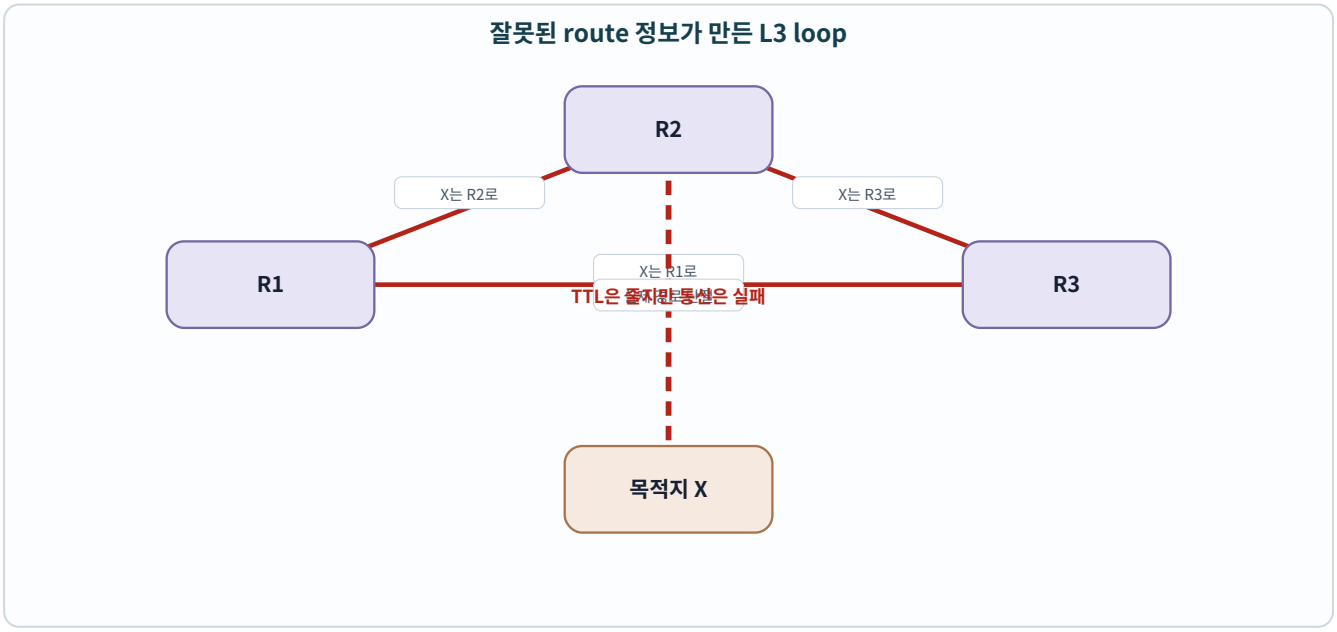


그림 14-2. L2 loop와 달리 IP packet은 TTL로 수명이 제한되지만 route가 수렴할 때까지 blackhole과 지연이 생긴다.

Stub network

다른 network로 가는 transit path가 아니라 연결 방향이 제한된 edge network를 뜻한다. 하나의 exit만 가진 LAN이 대표적이다. 기술 분야의 stub은 완전한 구현 대신 interface만 제공하는 시험용 코드, 실제 대상을 대신하는 작은 대체물이라는 의미로도 쓰인다.

15장. RIP를 통해 배우는 Distance Vector와 수렴 문제

> RIP은 단순해서 교육용으로만 보이지만, “이웃이 알려준 거리를 믿고 경로를 갱신한다”는 distance vector의 장점과 한계를 가장 선명하게 보여준다. count-to-infinity, poison, split horizon, timer, NBMA 문제를 하나의 장애 흐름으로 연결한다.

RIP와 Distance Vector

필기 확인 + 보충 해설

전체 지도를 모르는 router가 이웃의 말을 이용하는 방식

함께 찾을 말: RIP, RIPv1, RIPv2, Distance Vector, Hop Count, Full Update, Auto Summary

RIP와 Distance Vector란 RIP은 neighbor가 광고한 목적지 hop count에 1을 더해 route를 선택하는 distance-vector IGP이며, 15 hop까지를 reachable로 취급한다.

Distance와 Vector

distance는 목적지까지의 비용, vector는 그 방향(next hop)을 뜻한다. router는 전체 topology map을 갖지 않고 neighbor가 광고한 destination과 metric에 자기 link 비용을 더해 최선 route를 선택한다. 이웃의 정보가 낡거나 서로 의존하면 loop가 생길 수 있다.

RIP metric과 infinity

RIP은 router hop count를 metric으로 사용한다. 유효 경로의 최대는 15 hop이고 16은 unreachable/infinity로 취급한다. 작은 infinity는 count-to-infinity가 영원히 증가하는 것을 제한하지만 큰 network에는 확장성이 부족하다.

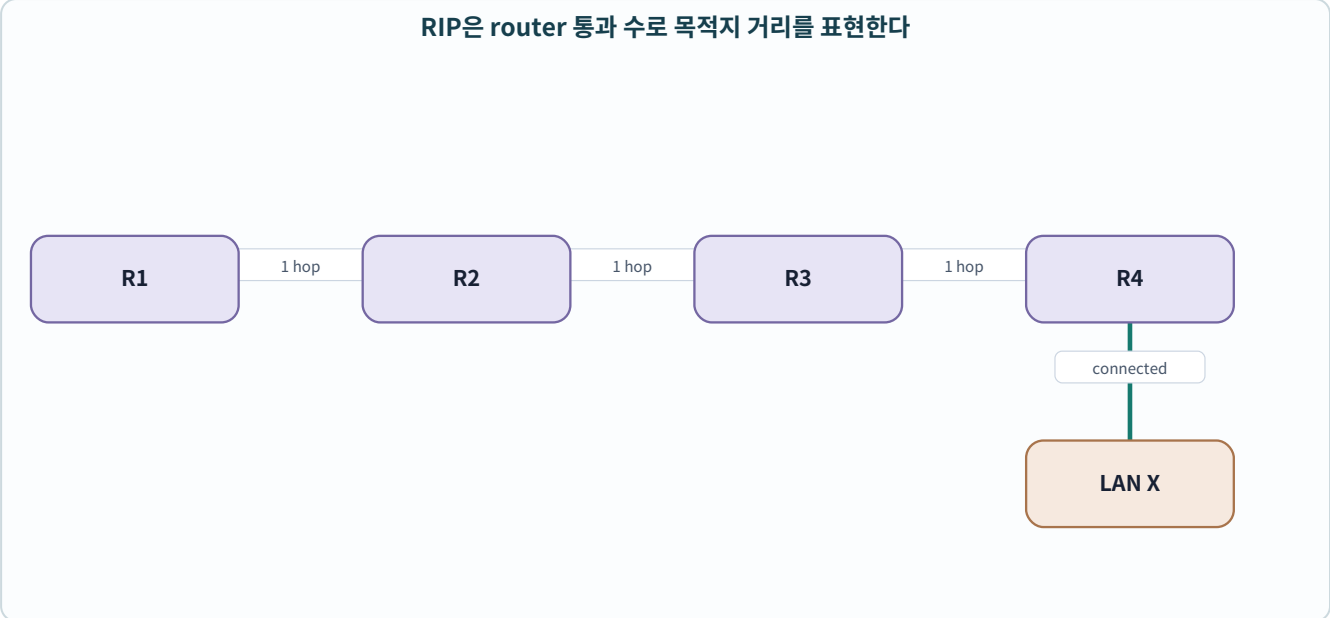


그림 15-1. 대역폭 10Gbps와 10Mbps 경로도 hop 수가 같으면 같은 metric일 수 있다.

RIPv1과 RIPv2

항목	RIPv1	RIPv2
prefix mask 광고	하지 않음(classful)	포함(classless, VLSM/CIDR)
update destination	broadcast	224.0.0.9 multicast

항목	RIPv1	RIPv2
인증	기본 지원 없음	인증 확장 가능
자동 summary	classful 경계 영향	설정으로 제어 가능

Full update와 periodic update

RIP은 기본적으로 일정 주기마다 routing table 정보를 neighbor에게 보낸다. 변화분만 보내는 link-state/advanced protocol과 비교해 full update라고 설명한다. triggered update는 중요한 변화가 생기면 정기 timer를 기다리지 않고 즉시 광고한다.

Count-to-Infinity · Split Horizon · Route Poisoning

필기 확인 + 보충 해설

RIP의 장애 방지 기능이 왜 각각 필요한가

함께 찾을 말: Count to Infinity, Split Horizon, Route Poisoning, Poison Reverse, Triggered Update, Hold-down

Count-to-Infinity · Split Horizon · Route Poisoning란 count-to-infinity는 distance-vector router가 서로의 낡은 route를 대체 경로로 믿어 metric을 계속 높이는 문제이고, split horizon · poisoning · triggered update · hold-down은 이를 줄이는 수렴 기법이다.

Count-to-Infinity가 시작되는 상황

R3 뒤의 network X가 끊겼는데 R2가 아직 “X는 R3 방향 1 hop”이라는 옛 route를 갖고 있고, R3가 R2의 광고를 듣고 “R2를 통해 2 hop”이라고 잘못 믿으면 서로 metric을 3, 4, 5로 올리며 loop가 지속된다.

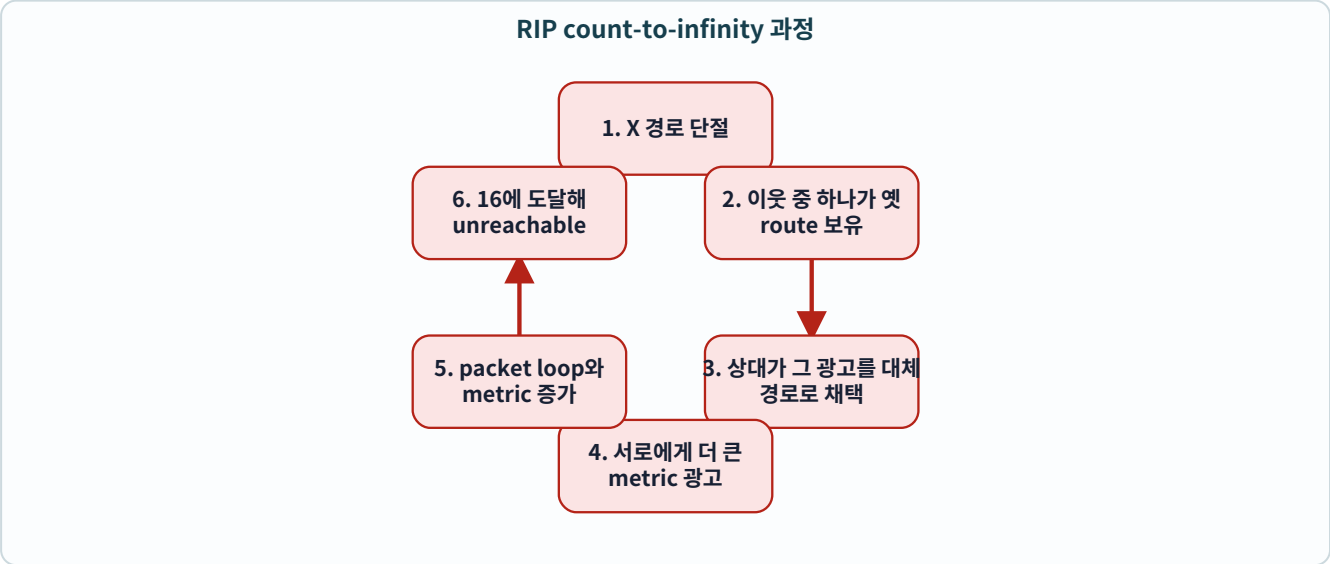


그림 15-2. 서로의 route가 사실 상대가 자신에게서 배운 정보라는 사실을 알지 못하는 것이 핵심이다.

Split Horizon

어떤 interface로 배운 route를 같은 interface 방향으로 다시 광고하지 않는다. “네가 나에게 알려준 길을 다시 네게 새 길처럼 말하지 않겠다”는 규칙이다. 두 router loop에 강하지만 multi-access와 hub-and-spoke 환경에서 필요한 spoke 간 route 전달을 막을 수 있다.

Route Poisoning과 Poison Reverse

route poisoning은 실패한 route를 metric 16으로 즉시 광고해 더 이상 쓰지 말라고 알린다. poison reverse는 split horizon 대신 또는 함께, 배운 방향으로 그 route를 metric 16으로 명시적으로 되돌려 상대가 자신을 next hop으로 선택하지 않게 한다.

Triggered Update와 Hold-down

triggered update는 장애 정보를 즉시 전파한다. hold-down은 실패 후 일정 기간 더 나빠진 정보나 의심스러운 대체 route를 받아들이지 않아 flapping과 loop를 줄이지만, 유효한 복구 경로 채택을 늦출 수 있다. 구현별 timer 동작을 확인해야 한다.

포이즈닝은 packet을 오염시키는 공격이 아니다

RIP 문맥의 route poisoning은 의도적으로 metric을 infinity로 광고해 route를 폐기시키는 정상 loop 방지 기법이다. DNS cache poisoning 같은 공격 용어와 구분한다.

RIP Timer·Offset List·Distribute List

필기 확인 + 보충 해설

수렴 속도와 경로 정책을 조정하는 운영 도구

함께 찾을 말: RIP Timers, Offset List, Distribute List, Passive Interface

RIP Timer·Offset List·Distribute List란 RIP timer는 update의 생명주기와 route 제거 시점을 제어하고, offset list는 metric을 조정하며, distribute list는 routing update의 prefix 수신·광고를 filter한다.

Timer의 역할

Timer	개념적 역할
Update	정기 route 광고 주기
Invalid/Timeout	일정 시간 갱신이 없으면 route를 invalid로 판단
Hold-down	불안정한 변경을 일정 시간 보수적으로 처리
Flush/Garbage collection	invalid route를 table에서 제거할 때까지 대기
Triggered update jitter 등	동시 update 폭주와 즉시 전파를 조절하는 구현 요소

정확한 숫자는 플랫폼을 확인한다

교재에서 Cisco RIP의 흔한 기본값으로 update 30초, invalid 180초, hold-down 180초, flush 240초를 배우지만 vendor와 구현, address family에 따라 다를 수 있다. `show ip protocols`로 실제 timer를 확인한다. “다섯 timer”라는 분류도 구현 설명 방식에 따라 다르다.

Offset List

특정 route의 inbound/outbound RIP metric에 값을 더해 경로 선호를 조정한다. hop count 조작은 topology를 바꾸지 않고 selection을 유도하지만 16을 넘겨 route를 unreachable하게 만들 수 있고 의도와 다른 backup path를 선택하게 할 수 있다.

Distribute List

routing update에서 특정 prefix를 수신·광고할지 filter하는 정책이다. packet ACL과 달리 user traffic packet 자체를 직접 막는 것이 아니라 routing information을 제어한다.

RIPv2 기본 확인 예시

```
router rip
 version 2
 no auto-summary
 network 10.0.0.0
 passive-interface default
 no passive-interface GigabitEthernet0/0

show ip protocols
show ip route rip
debug ip rip        ! 실습에서만, CPU·로그 부하 주의
```

NBMA 환경과 RIP

필기 확인 + 보충 해설

"같은 subnet"이어도 모든 neighbor가 broadcast를 받는 것은 아니다

함께 찾을 말: NBMA, Non-Broadcast Multiple Access, Hub and Spoke, BMA

NBMA 환경과 RIP란 NBMA는 여러 endpoint가 한 network를 공유하지만 native broadcast 전달이 없는 구조이며, RIP은 update 전달과 split horizon 때문에 hub-and-spoke에서 추가 설정이 필요하다.

NBMA란

Non-Broadcast Multiple Access는 여러 endpoint가 하나의 L3 network를 공유하지만 Ethernet처럼 native broadcast/multicast를 모든 peer에게 자동 전달하지 않는 network 유형이다. Frame Relay가 전통적 예다.

RIP과 NBMA의 충돌

RIP update가 broadcast/multicast를 전제로 하는데 NBMA cloud가 이를 peer마다 복제하지 않으면 neighbor가 update를 받지 못한다. hub에서 split horizon이 켜져 있으면 한 spoke에서 배운 route를 같은 hub interface의 다른 spoke에 광고하지 않아 spoke 간 reachability가 끊길 수 있다.

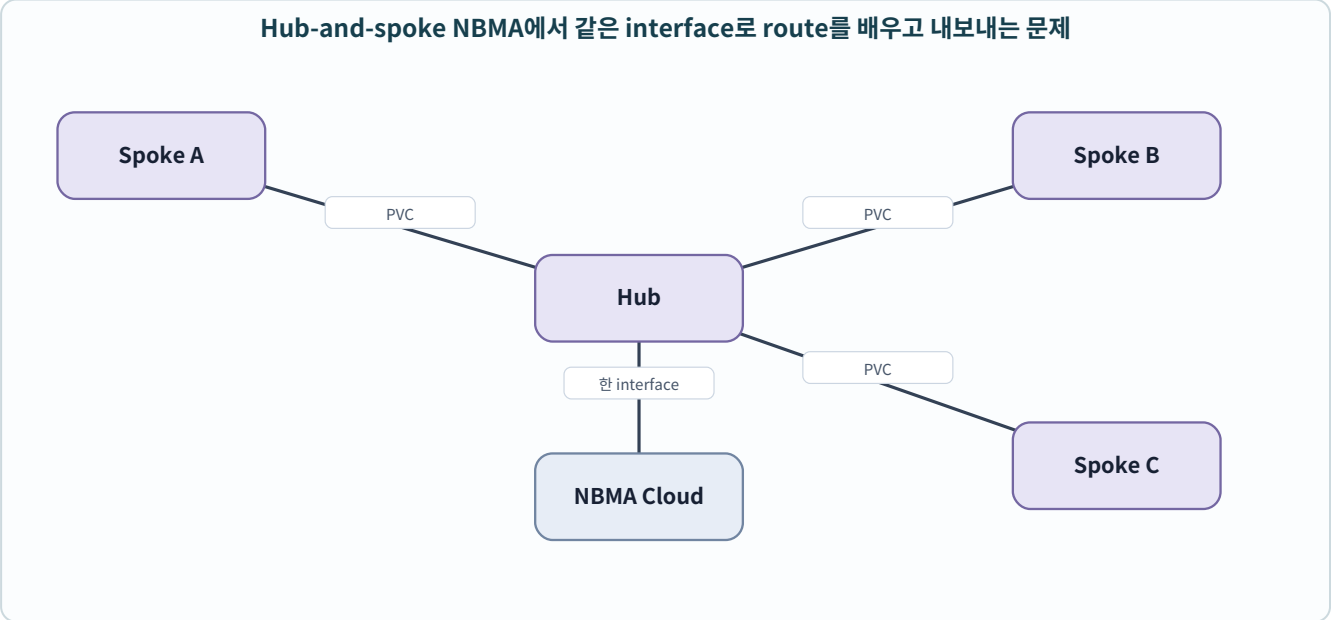


그림 15-3. split horizon 정책, point-to-point subinterface, neighbor 설정, multicast replication을 topology에 맞게 설계한다.

해결 방향

- hub의 해당 interface에서 split horizon을 끄되 loop 위험과 protocol 요구를 검토한다.
- spoke별 point-to-point subinterface로 논리 interface를 나눈다.
- unicast neighbor를 명시하거나 provider/overlay의 multicast·broadcast 전달 기능을 사용한다.
- 현대 WAN에서는 topology와 protocol에 맞는 다른 routing 설계를 선택한다.

16장. OSPF·EIGRP·BGP: 서로 다른 방식으로 경로를 합의한다

> Dynamic routing protocol은 모두 “route를 자동으로 배운다”는 공통점이 있지만, 어떤 정보를 교환하고 loop를 어떻게 피하며 경로를 무엇으로 평가하는지가 다르다. link-state, advanced distance vector, path vector를 비교한다.

IGP·EGP·AS

필기 확인 + 보충 해설

protocol을 조직 경계와 정책 범위로 분류한다

함께 찾을 말: IGP, EGP, Interior, Exterior, Autonomous System, ASN

IGP·EGP·AS란 IGP는 하나의 관리 domain 내부 routing을, EGP는 서로 다른 autonomous system 사이의 routing을 담당하며, AS는 공통 routing policy 아래 운영되는 network 집합이다.

IGP와 EGP

IGP는 하나의 administrative domain/AS 내부에서 경로를 교환하는 protocol 계열이며 RIP, OSPF, IS-IS, EIGRP가 대표적이다. EGP는 AS 사이의 정책 기반 경로 교환에 쓰이며 현재 사실상 BGP가 표준이다. interior/exterior는 조직 경계와 정책 책임을 나타낸다.

Autonomous System과 AS Number

AS는 하나의 일관된 routing policy 아래 Internet에 route를 광고하는 network 집합이고 ASN은 이를 식별하는 번호다. 사내 IGP area와 BGP AS는 같은 경계 개념이 아니다.

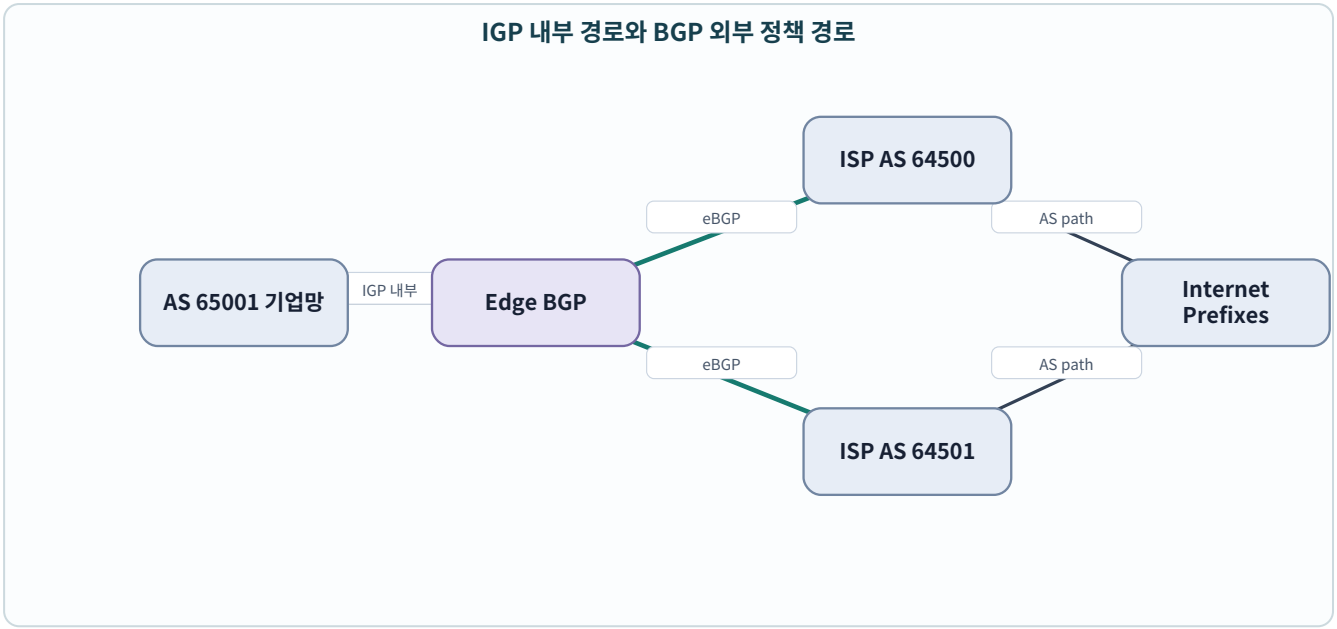


그림 16-1. AS 경계에서는 단순 최단거리보다 사업·보안·traffic engineering policy가 중요하다.

OSPF·Link State·Dijkstra

필기 확인 + 보충 해설

이웃의 거리만 듣지 않고 topology graph를 함께 가진다

함께 찾을 말: OSPF, Link State, Dijkstra, SPF, LSA, LSDB, Area, DR, BDR

OSPF·Link State·Dijkstra란 OSPF는 link 상태를 area 안에 flooding해 공통 topology database를 만들고, 각 router가 Dijkstra SPF로 최단 경로를 계산하는 link-state IGP다.

Link State의 핵심

각 router가 자신의 link와 neighbor 상태를 LSA로 알리고, area 안의 router들이 같은 link-state database를 만든다. 각 router는 그 topology graph에 Dijkstra SPF algorithm을 실행해 자신을 root로 한 shortest-path tree를 계산한다.

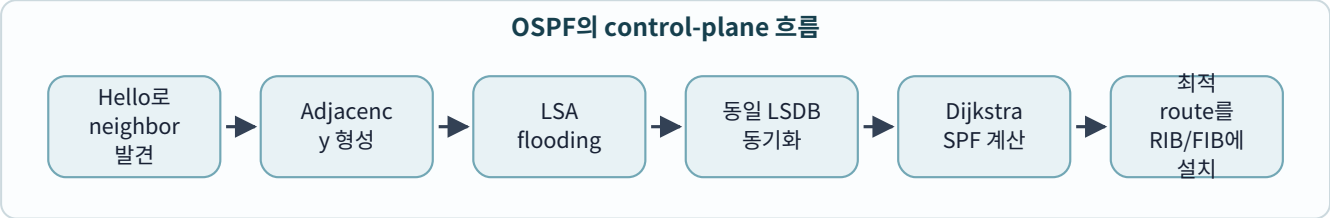


그림 16-2. route만 전달하는 것이 아니라 topology 정보를 공유해 각 router가 독립 계산한다.

Cost와 Area

OSPF metric은 interface cost의 합이다. 흔히 reference bandwidth / interface bandwidth로 계산하지만 vendor 설정과 reference 값에 따라 달라진다. area는 LSDB와 flooding 범위를 계층화하고 backbone area 0을 중심으로 연결한다.

DR/BDR

broadcast multi-access network에서 모든 router가 서로 full adjacency를 맺으면 관계 수가 급증한다. DR과 BDR을 선출해 LSA 교환 중심점을 만들고 adjacency 수를 줄인다. DR은 traffic forwarding gateway가 아니라 control-plane 역할이다.

Link state와 interface state

link state는 물리 carrier up/down만이 아니라 protocol 관점의 neighbor, cost, network type, LSA 상태를 포함할 수 있다. interface가 up/up이어도 OSPF authentication, area, timer, MTU mismatch로 adjacency가 안 될 수 있다.

EIGRP·DUAL·Successor·SIA

필기 확인 + 보충 해설

경로가 끊긴 뒤 query가 어디까지 퍼지는가

함께 찾을 말: EIGRP, DUAL, Successor, Feasible Successor, FD, Feasible Distance, RD, Reported Distance, K values, Stuck in Active, SIA

EIGRP·DUAL·Successor·SIA란 EIGRP는 composite metric과 DUAL algorithm으로 loop-free 경로를 계산하고, successor를 주 경로, feasibility condition을 만족한 feasible successor를 즉시 사용 가능한 backup으로 유지한다.

EIGRP metric과 K 값

EIGRP classic metric은 기본적으로 minimum bandwidth와 cumulative delay를 사용하며 K1·K3가 활성화된다. reliability와 load를 반영하는 K2, K4, K5는 값이 빠르게 변해 route flapping과 계산 불안정을 만들 수 있고 neighbor의 K 값이 맞아야 하므로 일반적으로 사용하지 않는다.

Successor와 Feasible Successor

successor는 topology table에서 선택된 최적 next hop, feasible successor는 현재 successor가 실패할 때 loop 없이 즉시 쓸 수 있다고 DUAL feasibility condition을 만족한 backup route다.

FD·RD와 feasibility condition

Reported Distance(RD)는 neighbor가 목적지까지 보고한 거리이고, Feasible Distance(FD)는 local router가 선택한 최적 경로의 총 거리다. 후보 neighbor의 RD가 현재 successor의 FD보다 작으면 그 neighbor는 나보다 목적지에 더 가깝다고 볼 수 있어 loop-free backup으로 인정한다. 간단히 RD < current FD다. “FD > RD”라는 암기는 어떤 경로의 값끼리 비교하는지 명시해야 한다.

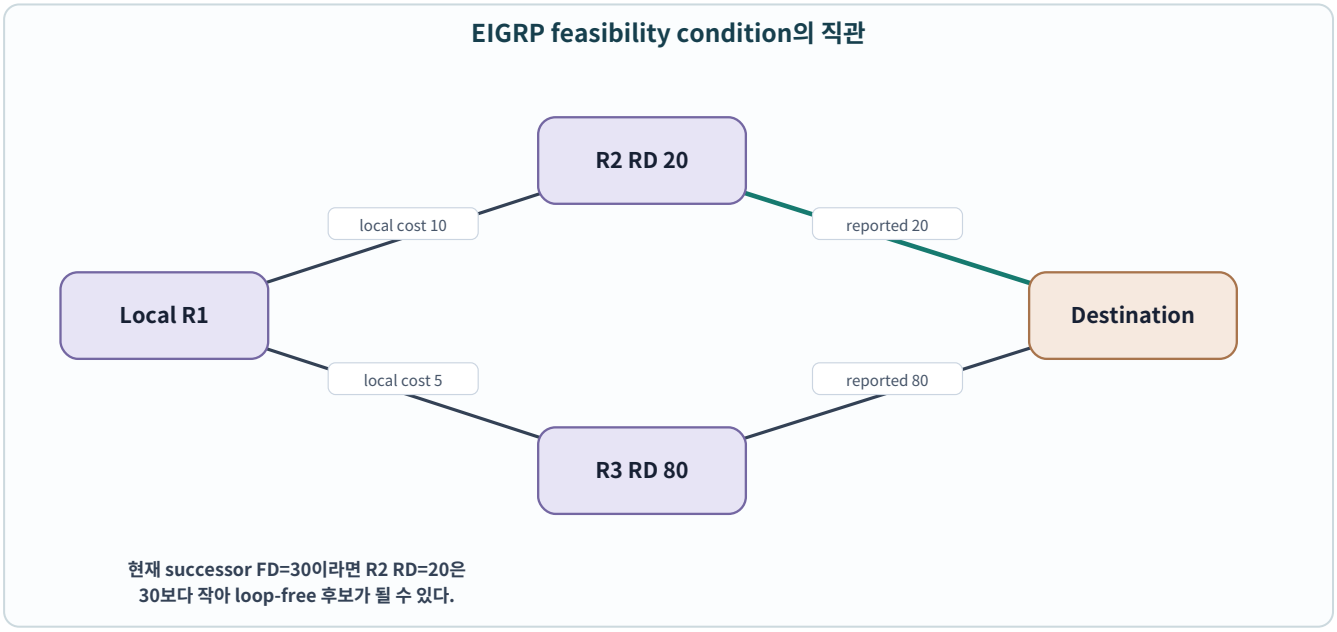


그림 16-3. RD는 neighbor 관점, FD는 local total 관점이다.

DUAL과 SIA

successor가 사라지고 feasible successor가 없으면 router는 route를 active 상태로 두고 neighbor에게 query한다. 모든 reply를 받아야 새 route를 확정한다. reply가 제한 시간 안에 오지 않으면 Stuck-in-Active가 되어 adjacency reset과 route disruption이 생길 수 있다. query boundary, summarization, stub 설정으로 범위를 줄인다.

EIGRP packet

Hello는 neighbor 유지, Update는 route 정보, Query는 대체 경로 질의, Reply는 질의 응답, ACK는 reliable EIGRP packet 확인에 쓰인다. protocol 번호와 RTP(Reliable Transport Protocol)를 사용하며 TCP의 RTP와 다른 이름이다.

BGP · Path Vector

필기 확인 + 보충 해설

Internet 경로는 단순한 최단거리보다 정책으로 결정된다

함께 찾을 말: BGP, Path Vector, AS_PATH, eBGP, iBGP, RPKI

BGP · Path Vector란 BGP는 AS 사이에서 prefix와 AS_PATH 등 path attribute를 교환하고 local policy에 따라 best route를 선택하는 path-vector routing protocol이다.

Path Vector

BGP는 destination prefix뿐 아니라 그 route가 지나온 AS_PATH와 여러 attribute를 광고한다. 자기 ASN이 path에 이미 있으면 loop로 판단해 거부할 수 있다. 단순 shortest path가 아니라 local preference, policy, AS path, origin, MED 등 단계별 정책으로 best path를 고른다.

eBGP와 iBGP

eBGP는 다른 AS 사이, iBGP는 같은 AS 내부 BGP speaker 사이의 route 전달이다. iBGP는 AS_PATH를 늘리지 않으므로 full mesh, route reflector, confederation 같은 loop 방지 구조가 필요하다.

BGP는 reachability와 policy protocol이다

Internet 규모에서 모든 link topology를 공유하지 않고 prefix별 path attribute를 전달한다. convergence가 IGP보다 느릴 수 있지만 route filtering과 사업 관계, traffic engineering을 표현하는 유연성이 중요하다.

BGP hijack의 원리

잘못되거나 악의적인 AS가 자신이 소유하지 않은 prefix 또는 더 구체적인 prefix를 광고하면 다른 network가 그 경로를 선택할 수 있다. prefix filter, RPKI route origin validation, monitoring이 방어 수단이다.

17장. Serial·PPP·Frame Relay·ACL·QoS: WAN과 경계 제어

> LAN이 Ethernet 중심이라면 전통 WAN 과정에서는 point-to-point serial과 provider cloud의 virtual circuit을 배운다. 오늘날 직접 쓰는 빈도는 줄어도 encapsulation, link negotiation, NBMA, access policy의 원리를 이해하는 데 유용하다.

Serial·HDLC·PPP·LCP/NCP·PAP/CHAP

필기 확인 + 보충 해설

회선이 물리적으로 연결된 뒤 L3 packet을 보내기까지

함께 찾을 말: Serial Interface, HDLC, PPP, LCP, NCP, PAP, CHAP, PPPoE

Serial·HDLC·PPP·LCP/NCP·PAP/CHAP란 PPP는 point-to-point link의 frame encapsulation과 협상·인증·L3 설정을 제공하며, LCP는 link, NCP는 network protocol을 구성하고 PAP/CHAP은 peer 인증 방식이다.

Serial과 Ethernet

serial은 bit를 순차적으로 보내는 넓은 개념이지만 network 교육에서는 router의 동기식 WAN serial interface와 전용선/Frame Relay 연결을 가리키는 경우가 많다. Ethernet은 MAC address와 frame format, multi-access 동작을 정의하는 LAN 계열 기술이다.

HDLC

HDLC는 bit-oriented data-link protocol 계열이다. Cisco serial interface의 기본 encapsulation인 Cisco HDLC는 표준 HDLC에 protocol type 정보를 추가한 vendor 변형이라 다른 vendor와 상호운용 시 PPP가 더 적합할 수 있다.

PPP와 LCP/NCP

PPP는 point-to-point link에서 encapsulation, link 설정, authentication, network-layer protocol 설정을 제공한다. LCP가 link parameter와 authentication을 협상하고, NCP가 IPv4 IPCP 등 각 L3 protocol을 설정한다.

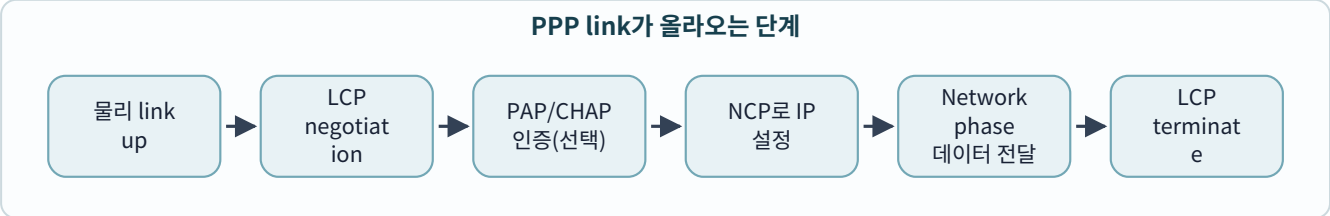


그림 17-1. 한 interface에서도 encapsulation 설정에 따라 HDLC 또는 PPP를 사용할 수 있다.

PAP와 CHAP

PAP는 username/password를 link에 반복 또는 초기 전송하는 단순 2-way 방식으로 암호화 자체가 없으면 노출 위험이 크다. CHAP는 challenge와 shared secret 기반 digest를 사용해 password 자체를 전송하지 않고 peer를 검증하며 주기적 재검증이 가능하다. 오래된 hash 설계와 shared secret 관리 한계는 별도 고려해야 한다.

Frame Relay·DLCI·PVC·Subinterface

필기 확인 + 보충 해설

물리 회선 한 개를 여러 논리 경로로 나누는 전통 WAN

함께 찾을 말: Frame Relay, DLCI, PVC, SVC, Point-to-point, Multipoint, Subinterface

Frame Relay·DLCI·PVC·Subinterface란 Frame Relay는 provider NBMA cloud 위에 virtual circuit을 제공하고, DLCI는 local access link에서 PVC를 식별하는 label이며, subinterface는 한 물리 interface를 논리 L3 interface로 나눈다.

Frame Relay의 virtual circuit

provider의 packet-switched NBMA cloud 위에 고객 router 사이의 logical PVC를 제공한다. 각 access link는 여러 PVC를 multiplex할 수 있고 local DLCI로 circuit을 식별한다. DLCI는 해당 local interface 의미라 양 끝에서 숫자가 반드시 같을 필요가 없다.

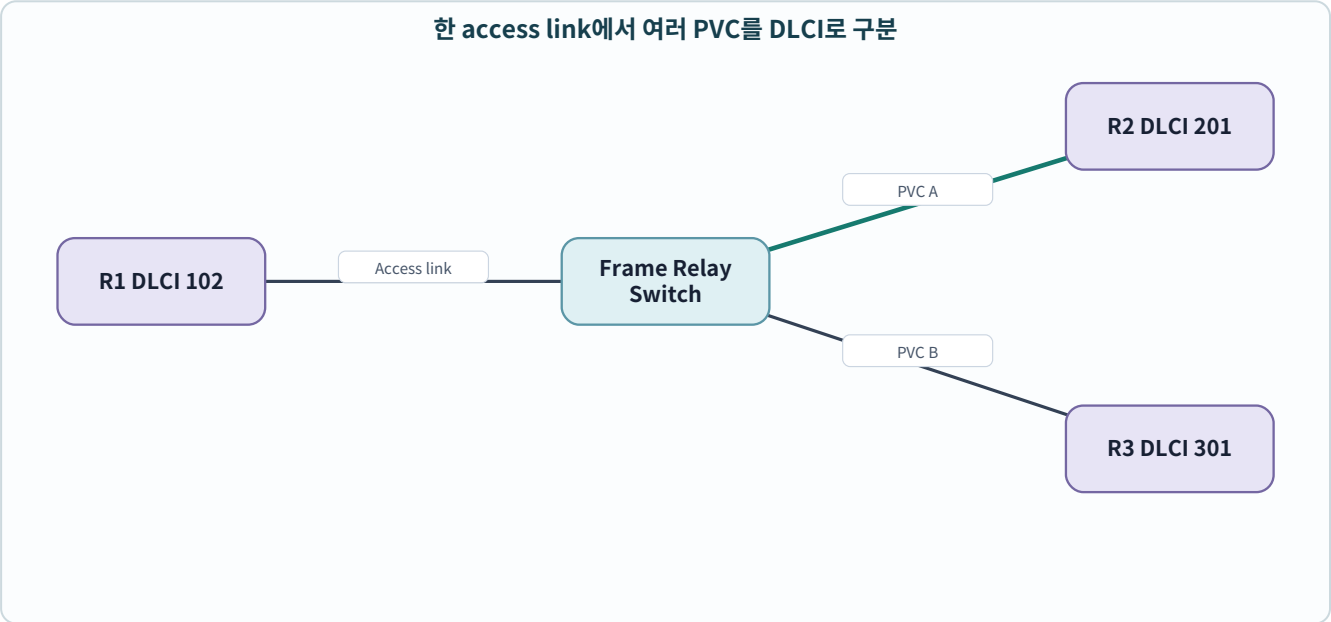


그림 17-2. DLCI는 provider cloud 안의 end-to-end global address가 아니라 local label이다.

Point-to-point와 Multipoint subinterface

point-to-point subinterface는 한 peer/PVC를 별도 logical interface와 subnet으로 다뤄 broadcast-like protocol 문제를 단순화한다. multipoint는 여러 peer를 한 subnet/interface에 두지만 inverse ARP, frame-relay map, split horizon, broadcast replication을 고려해야 한다.

PVC와 SVC

PVC는 관리적으로 상시 설정된 permanent virtual circuit이고, SVC는 필요할 때 signaling으로 설정되는 switched virtual circuit이다. 교육 실습은 PVC 중심이다.

ACL·Permit/Deny·Wildcard Mask·Inbound/Outbound

필기 확인 + 보충 해설

rule 내용만큼 순서와 적용 interface·방향이 중요하다

함께 찾을 말: Access Control List, ACL, Permit, Deny, Implicit Deny, Wildcard Mask, Inbound, Outbound

ACL·Permit/Deny·Wildcard Mask·Inbound/Outbound란 ACL은 packet header 조건을 순서대로 검사해 첫 일치 rule의 permit/deny를 적용하는 접근 제어 목록이며, wildcard mask는 주소 비교에서 검사할 bit와 무시할 bit를 표시한다.

ACL은 순서 있는 조건 목록이다

packet의 source/destination address, protocol, port 등 field를 위에서 아래로 비교해 처음 일치한 permit/deny를 적용한다. 끝에는 일반적으로 명시되지 않은 deny any가 있다고 이해한다. 따라서 넓은 rule을 먼저 두면 아래의 세부 rule이 절대 평가되지 않을 수 있다.

Standard와 Extended

Cisco IPv4 standard ACL은 주로 source address만, extended ACL은 source/destination, protocol, port 등 더 많은 조건을 검사한다. standard는 destination 가까이, extended는 source 가까이 배치한다는 교육 원칙은 불필요 traffic 차단과 오차단 영향 범위를 줄이기 위한 일반 지침이며 topology와 관리 정책에 따라 검토한다.

Wildcard mask

Cisco ACL·OSPF network statement 등의 wildcard mask에서 0bit는 해당 address bit가 반드시 일치해야 하고 1bit는 무시한다. 연속 mask에서는 subnet mask를 255에서 뺀 값과 대응하지만 discontiguous wildcard도 표현할 수 있어 단순 “역 subnet mask”라고만 보면 한계가 있다.

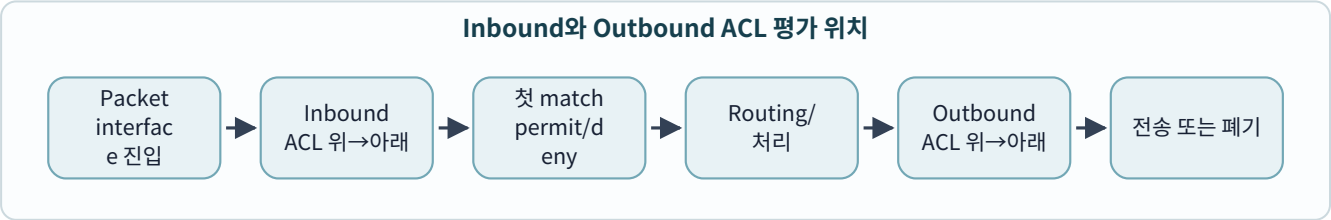


그림 17-3. 한 packet이 모든 interface ACL을 무조건 거치는 것이 아니라 실제 ingress/egress 경로의 적용 지점에서 평가된다.

Cisco ACL 예시

```
ip access-list extended WEB-IN
 10 permit tcp 10.10.0.0 0.0.255.255 host 192.0.2.10 eq 443
 20 deny ip any host 192.0.2.10 log
 30 permit ip any any
!
interface GigabitEthernet0/1
 ip access-group WEB-IN in

show access-lists WEB-IN
show ip interface GigabitEthernet0/1
```

"작은 범위부터/큰 범위부터"의 정답

ACL은 첫 일치가 끝이므로 **예외적이고 구체적인 rule을 먼저**, 넓은 일반 rule을 뒤에 두는 것이 기본이다. 다만 policy 목표를 먼저 문장으로 쓰고 test packet matrix로 순서를 검증해야 한다.

QoS

필기 확인 + 보충 해설

대역폭을 늘리는 기술이 아니라 혼잡 시 누구를 먼저 처리할지 정한다

함께 찾을 말: Quality of Service, QoS, DSCP, PCP, Policing, Shaping, Scheduling

QoS란 QoS는 제한된 network 자원에서 traffic을 분류·표시하고 queue와 rate를 제어해 지연·손실·대역폭 요구가 다른 application에 차등 서비스를 제공하는 정책 체계다.

QoS가 필요한 이유

대역폭이 부족하거나 순간 burst가 queue를 만들 때 voice, control, bulk backup을 같은 방식으로 처리하면 지연에 민감한 traffic이 손상된다. QoS는 traffic을 분류하고 표시하며 queue·scheduling·policing·shaping 정책으로 제한된 자원을 목적에 맞게 배분한다.

Classification·Marking

5-tuple, DSCP, interface, application 정보로 traffic class를 나누고 IP DSCP나 Ethernet 802.1p PCP에 우선순위를 표시한다. marking은 신뢰 경계에서 검증해야 하며 client가 임의로 높은 우선순위를 붙인 것을 그대로 믿으면 안 된다.

Policing과 Shaping

policing은 설정 rate를 넘는 traffic을 drop하거나 remark해 즉시 제한하고, shaping은 buffer에 보관해 평균 전송률을 맞춘다. shaping은 지연을 추가하지만 burst drop을 줄일 수 있다.

Scheduling

strict priority queue는 지연 민감 traffic을 먼저 처리하지만 과도하면 다른 class를 굶길 수 있다. weighted fair/round-robin 계열은 class별 비율을 보장한다. 실제 algorithm과 명칭은 vendor/platform마다 다르다.

PART IV

서버·스토리지·웹·DB·DNS 운영

패킷이 서버에 도착한 뒤에는 운영체제, process, filesystem, web server, application runtime, database가 차례로 요청을 처리한다. 이 부분은 설치 명령을 나열하는 대신 각 구성요소가 왜 분리되며 장애와 보안 경계가 어디에 생기는지를 설명한다.

18장. 서버 하드웨어와 부팅: Rack·Blade·BMC·디스크가 협력하는 법

서버는 “성능 좋은 PC”만을 뜻하지 않는다. 원격 관리, 중복 전원, hot-swap disk, ECC memory, RAID controller, rack 배치처럼 장애가 나도 service를 유지하고 교체·관제하기 위한 기능이 강조된다.

Rack Server·Blade Server·BMC/iLO

필기 확인 + 보충 해설

장애가 난 OS 밖에서 서버를 관리하는 하드웨어 경계

함께 찾을 말: Server Rack, Blade Server, BMC, iLO, Out-of-band Management, SPU

Rack Server·Blade Server·BMC/iLO란 rack server는 표준 rack에 장착하는 독립 서버이고, blade server는 chassis 자원을 공유하는 modular compute이며, BMC/iLO는 OS와 독립적으로 전원·sensor·console을 관리하는 processor다.

Rack server와 server rack

rack은 표준 폭과 U 높이 단위로 서버·switch·PDU를 수직 장착하는 구조물이다. rack server는 독립 chassis가 rack에 들어가는 형태다. cable, airflow, 전원 A/B feed, 무게와 열 배치를 함께 설계해야 한다.

Blade server

blade chassis가 전원, cooling, management, network fabric을 공유하고 얇은 compute blade를 꽂는 형태다. 밀도와 중앙 관리가 장점이지만 chassis 장애영역과 vendor 종속, 초기 비용을 고려한다.

BMC와 iLO

BMC(Baseboard Management Controller)는 main CPU와 OS와 별도로 동작해 전원 on/off, sensor, remote console, virtual media를 제공하는 관리 processor다. HPE iLO는 BMC 제품군 이름이다. OS가 crash해도 out-of-band 관리가 가능하지만 management network와 credential이 공격되면 물리 console 수준의 통제권을 잃을 수 있다.

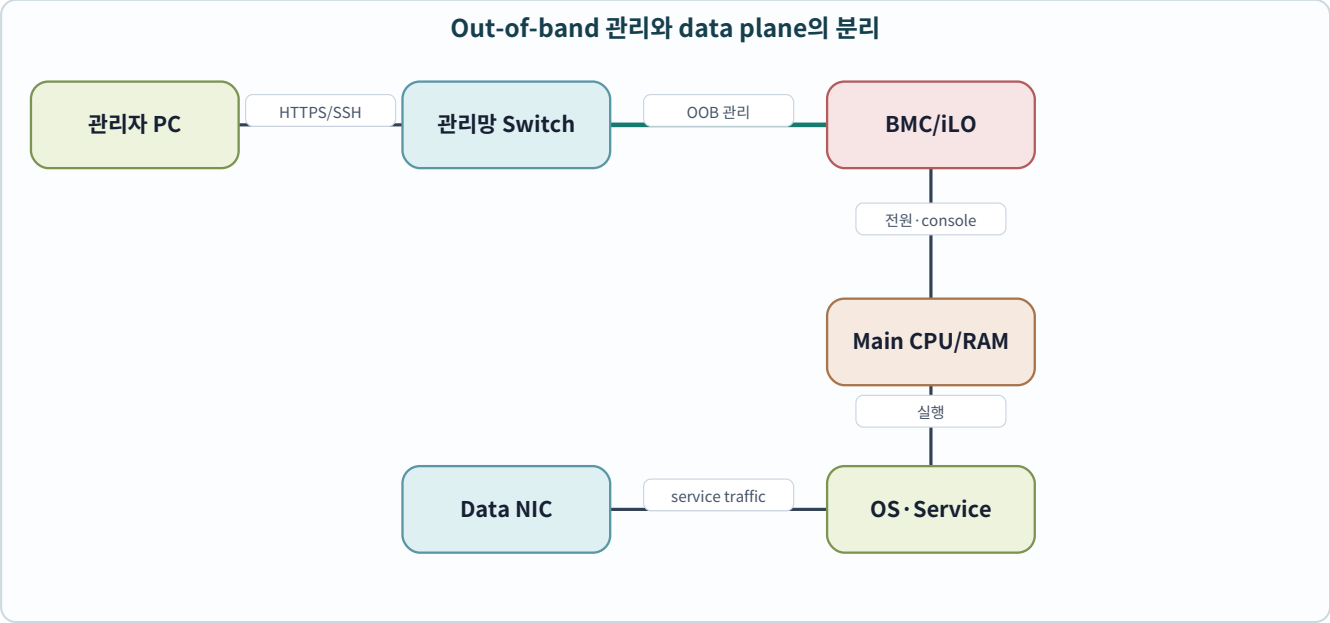


그림 18-1. BMC는 OS와 독립 경로이므로 별도 VLAN, MFA, ACL, firmware 관리가 필요하다.

SPU라는 약어

SPU는 vendor와 문맥에 따라 Security Processing Unit, Storage Processing Unit, Signal Processing Unit 등 여러 뜻으로 쓰인다. 필기에서 단어만 있을 때는 특정 제품 문서나 수업 장비를 확인해야 한다. 교재에서는 “특정 작업을 CPU에서 분리해 가속하는 전용 processor”라는 공통 개념으로만 다룬다.

서버 부팅·UEFI·MBR·GPT·Partition

필기 확인 + 보충 해설

전원 버튼에서 systemd service까지 이어지는 chain

함께 찾을 말: Boot Process, BIOS, UEFI, Bootloader, MBR, GPT, Primary Partition, Extended Partition, Sector, Cylinder

서버 부팅·UEFI·MBR·GPT·Partition란 서버 부팅은 firmware가 hardware를 초기화하고 bootloader가 kernel을 적재해 init process를 시작하는 과정이며, partition table은 disk 공간의 논리 구간을 기록한다.

전원이 들어온 뒤의 큰 흐름

1. 전원과 BMC가 hardware 상태를 확인한다.
2. CPU reset vector에서 firmware(UEFI/BIOS)가 시작된다.
3. POST와 memory·device 초기화 후 boot device를 선택한다.
4. UEFI는 EFI System Partition의 bootloader를, legacy BIOS는 MBR boot code를 읽는 식으로 bootloader를 실행한다.
5. bootloader가 kernel과 initramfs를 memory에 적재하고 kernel로 제어를 넘긴다.
6. kernel이 driver·root filesystem을 준비하고 PID 1(systemd 등)을 실행한다.
7. service unit과 login 환경이 올라온다.

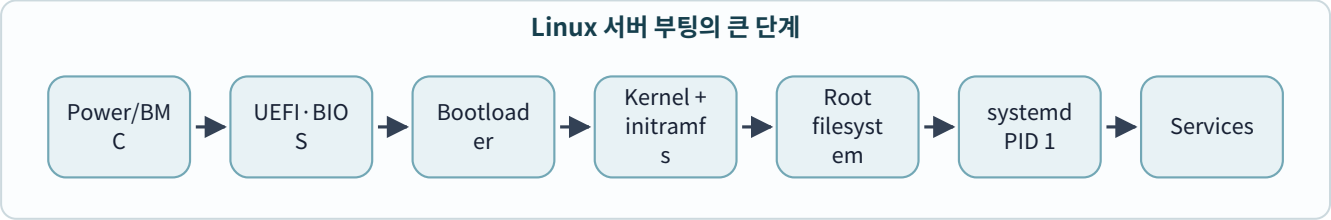


그림 18-2. 어느 단계에서 실패했는지에 따라 console에 보이는 증상과 복구 도구가 달라진다.

MBR과 GPT

MBR은 disk 첫 sector에 partition table과 boot code를 두는 오래된 방식으로 primary partition 수와 disk 크기 한계가 있다. extended partition 안에 logical partition을 두어 수를 늘렸다. GPT는 GUID로 partition을 식별하고 큰 disk와 많은 partition, backup table, CRC를 제공하며 UEFI와 함께 일반적이다.

Sector와 cylinder

sector는 disk가 block 단위로 읽고 쓰는 기본 주소 단위이며 전통적으로 512byte, 현대 disk는 4KiB physical sector가 흔하다. cylinder/head/sector는 회전 disk의 물리 기하를 표현하던 모델이고 현대 LBA에서는 연속 block 번호로 추상화한다.

19장. Disk·RAID·Filesystem·LVM: 데이터가 저장되는 여러 층

> “디스크를 mount한다”는 한 문장 안에는 block device, partition, RAID, volume manager, filesystem, directory namespace가 겹쳐 있다. 장애 대응에서는 어느 층이 고장 났는지 구분해야 한다.

Block Device·Local/Remote Storage·SCSI/NVMe

필기 확인 + 보충 해설

파일 경로 아래에 존재하는 저장 계층

함께 찾을 말: Block Device, HDD, SSD, NVMe, IDE, SCSI, SAS, NAS, SAN, iSCSI, NFS, SMB

Block Device·Local/Remote Storage·SCSI/NVMe란 block device는 고정 크기 block 주소로 읽고 쓰는 저장장치 추상화이고, local·SAN·NAS·object storage는 접근 위치와 interface 단위가 다르다.

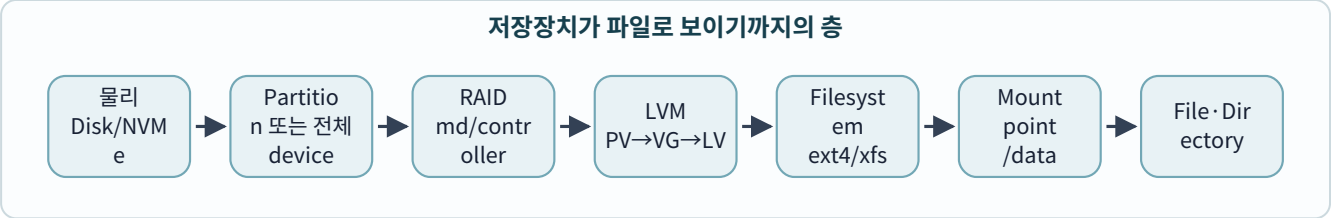


그림 19-1. 모든 환경이 모든 층을 쓰는 것은 아니지만 이름을 분리하면 명령과 장애 위치가 보인다.

Block device 이름

Linux의 SATA/SCSI disk는 `/dev/sda`, partition은 `/dev/sda1`처럼 보일 수 있고, NVMe는 `/dev/nvme0n1`, partition은 `p1` 접미를 쓴다. 이름은 탐지 순서에 따라 바뀔 수 있어 fstab은 UUID나 filesystem label을 쓰는 편이 안정적이다.

IDE·SCSI·SAS·NVMe

IDE/PATA는 오래된 parallel storage interface, SATA는 serial consumer/server disk interface, SCSI는 command set과 enterprise storage 계열, SAS는 serial attached SCSI, NVMe는 PCIe 기반 flash storage를 위한 queue 중심 protocol이다. Linux에서 많은 device가 SCSI mid-layer를 통해 보이더라도 물리 interface와 command model을 구분한다.

Local과 Remote Storage

분류	예	특징
직접/로컬 block	SATA SSD, NVMe, USB disk	host가 block을 직접 관리
SAN/block network	iSCSI, Fibre Channel	remote LUN을 local block처럼 사용
NAS/file network	NFS, SMB/CIFS	server가 filesystem을 관리하고 file protocol 제공
Object storage	S3 계열	object key와 API로 접근

RAID

필기 확인 + 보충 해설

여러 disk를 하나처럼 보이게 하지만 backup은 아니다

함께 찾을 말: RAID 0, RAID 1, RAID 5, RAID 6, RAID 10, RAID 01, Striping, Mirroring, Parity, Hot Swap

RAID란 RAID는 여러 disk에 striping·mirroring·parity를 조합해 성능, 용량, 장애 허용을 제공하는 block storage 구성 방식이다.

Striping·Mirroring·Parity

Striping

data block을 여러 disk에 나눠 병렬 처리. 성능·용량은 늘지만 redundancy가 없음.

Mirroring

같은 data를 여러 disk에 복제. 읽기·복구에 유리하지만 용량 효율이 낮음.

Parity

data에서 계산한 복구 정보를 분산 저장. disk 장애 시 남은 data와 parity로 재구성.

Level	최소 disk	가용 용량 개념	장애 허용·특징
RAID 0	2	합계	허용 없음. 한 disk 실패면 전체 array 손상
RAID 1	2	mirror 한 벌	mirror 구성에 따라 disk 실패 허용
RAID 5	3	N-1	1 disk parity, rebuild 중 위험·write penalty
RAID 6	4	N-2	2 disk parity
RAID 10	4	mirror 후 stripe	성능·복구 균형, 각 mirror pair 조건
RAID 01	4	stripe set을 mirror	RAID10과 장애 특성 다름

Hardware RAID와 Software RAID

hardware RAID controller는 cache, battery/flash-backed write protection, management를 제공할 수 있고 OS에는 logical disk로 보인다. Linux mdadm software RAID는 host CPU와 kernel md layer로 array를 구성한다. controller cache 정책과 disk write cache가 전원 장애 시 데이터 안전성에 중요하다.

Hot swap과 Cold swap

hot-swap은 system을 끄지 않고 교체하도록 backplane, controller, OS가 지원하는 기능이다. connector가 물리적으로 꽂힌다고 안전한 hot-swap은 아니다. cold-swap은 전원을 끄고 교체한다. rebuild가 끝날 때까지 array는 취약 상태일 수 있다.

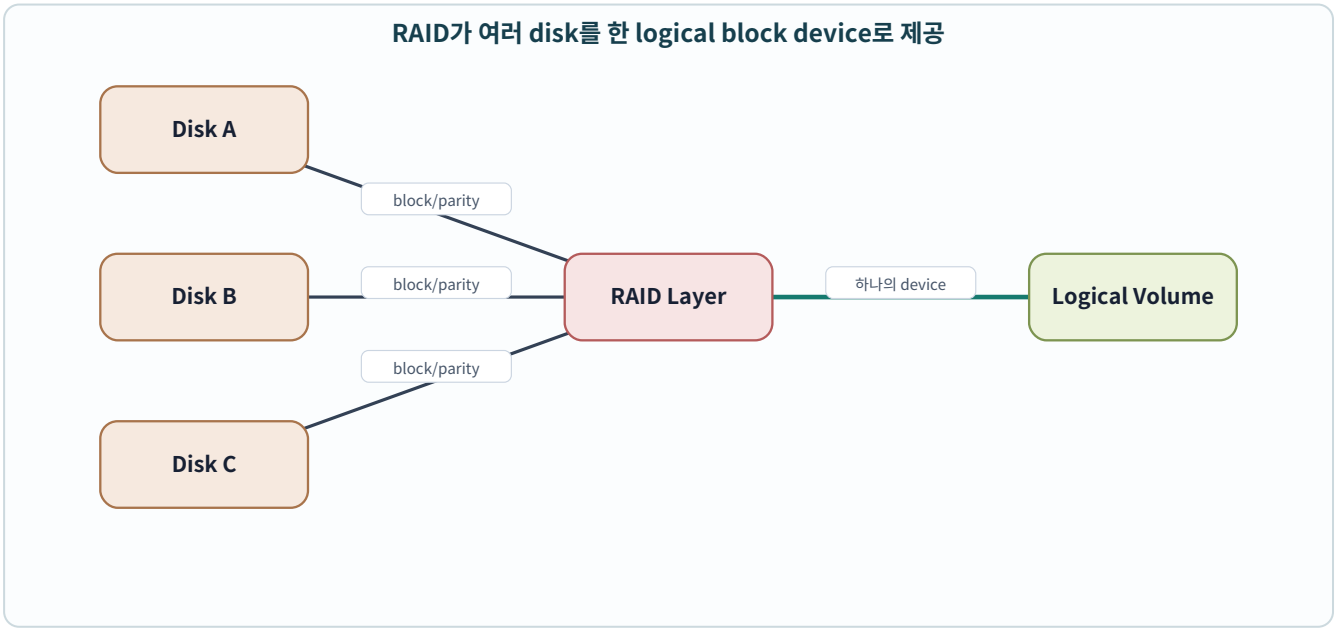


그림 19-2. RAID는 가용성을 높일 뿐, 삭제·암호화·site 장애를 막는 backup이 아니다.

RAID ≠ Backup

사용자 실수, ransomware, filesystem corruption, controller bug, 화재는 RAID 복제본에도 동시에 반영될 수 있다. version·offline/immutable·offsite backup과 복구 시험이 별도로 필요하다.

Filesystem·Inode·Journaling·CoW

필기 확인 + 보충 해설

용량이 남아도 파일을 못 만드는 이유까지 설명하는 구조

함께 찾을 말: Filesystem, ext4, XFS, JFS, Inode, Inode Exhaustion, Journaling, lost+found, Copy-on-Write, CoW

Filesystem·Inode·Journaling·CoW란 filesystem은 block device에 file·directory·metadata를 배치하는 규칙이고, inode는 file metadata와 data 위치를 저장하며, journaling과 CoW는 crash 일관성과 변경 기록 방식을 개선한다.

Filesystem이 하는 일

raw block에 file 이름, directory tree, metadata, free-space map, permission, timestamp를 저장하는 규칙이다. `mkfs`는 block device 위에 filesystem 구조를 초기화하고 기존 data를 덮을 수 있으므로 대상 확인이 중요하다.

Inode

Unix-like filesystem에서 inode는 file type, permission, owner, size, timestamp, data block 위치 등 metadata를 가진다. directory entry가 이름을 inode 번호에 연결한다. file 이름은 inode 안에 직접 저장되지 않는다는 설명이 핵심이다.

Inode exhaustion

작은 file을 매우 많이 만들면 data block 여유가 있어도 inode를 모두 사용해 새 file 생성이 실패할 수 있다. `df -h`는 용량, `df -i`는 inode 사용량을 본다. filesystem에 따라 inode 관리 방식은 다르다.

Journaling과 lost+found

journaling filesystem은 metadata 변경 의도를 journal에 먼저 기록해 crash 후 일관성 복구 시간을 줄인다. 모든 application data가 자동으로 transaction 안전해지는 것은 아니다. fsck가 연결을 잃은 inode를 복구할 때 ext 계열의 lost+found에 배치할 수 있다.

Copy-on-Write

CoW는 기존 block을 덮기 전에 새 위치에 변경 data를 쓰고 metadata pointer를 바꾸는 방식이다. snapshot·checksum과 잘 어울리지만 fragmentation, write amplification, free-space 관리 특성이 있다. Btrfs, ZFS 등에서 활용된다.

Filesystem과 mount 상태 확인

```
lsblk -f
blkid
df -hT
df -i
sudo fdisk -l
sudo mkfs.ext4 /dev/sdb1   # 주의: 대상 데이터 삭제
sudo mount /dev/sdb1 /mnt/data
findmnt /mnt/data
```

Mount·fstab·LVM·Quota

필기 확인 + 보충 해설

저장 공간을 만들고 운영 정책에 연결하는 단계

함께 찾을 말: mount, /etc/fstab, UUID, LVM, PV, VG, LV, Disk Quota, Soft Limit, Hard Limit, mdadm, metadisk

Mount·fstab·LVM·Quota란 mount는 filesystem을 directory tree에 연결하고, fstab은 그 관계를 선언하며, LVM은 block storage를 PV·VG·LV 계층으로 pool화하고, quota는 사용자별 사용량을 제한한다.

Mount의 의미

mount는 filesystem의 root를 Linux의 하나인 directory tree의 특정 directory에 연결하는 작업이다. mount point 아래에 원래 file이 있어도 mount된 동안 가려질 뿐 삭제된 것은 아니다. unmount하면 다시 보인다.

/etc/fstab과 UUID

fstab은 boot 또는 관리 명령에서 어떤 device를 어디에 어떤 filesystem·option으로 mount할지 선언한다. device name은 순서 변화가 있어 UUID를 쓰며, 문법 오류나 존재하지 않는 device는 boot 지연·emergency mode를 만들 수 있다. `mount -a`로 사전 검증한다.

LVM

Physical Volume(PV)을 Volume Group(VG)이라는 pool에 넣고 Logical Volume(LV)을 원하는 크기로 할당한다. disk와 partition 경계를 넘어 유연하게 확장하고 snapshot을 만들 수 있지만, filesystem 확장과 LV 확장은 별도 단계일 수 있고 축소는 filesystem 지원과 순서를 주의해야 한다.

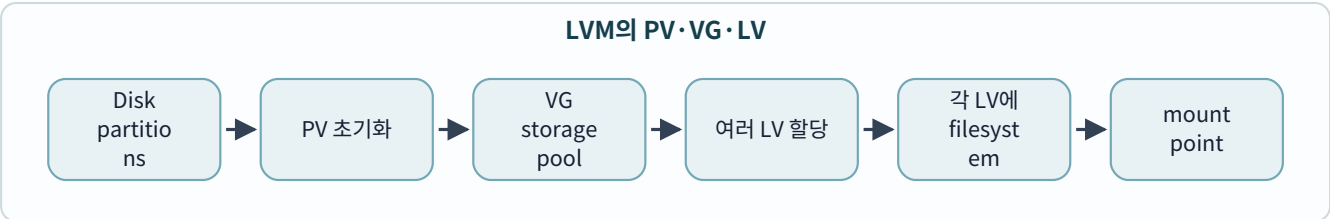


그림 19-3. LVM은 block allocation layer이고 filesystem 내용과 directory namespace는 그 위의 별도 층이다.

Disk quota

user/group/project가 사용할 block 용량과 file 수를 제한한다. soft limit은 grace period 동안 초과를 허용하고, hard limit은 즉시 넘지 못하게 한다. quota는 전체 disk가 한 사용자에게 소진되는 것을 막는 운영 통제다.

Mount와 LVM 예시

```
# fstab 예
UUID=1111-2222  /data  ext4  defaults,noexec,nosuid,nodev  0  2

# 검증
sudo mount -a
findmnt --verify

# LVM
pvs; vgs; lvs
sudo pvcreate /dev/sdb1
sudo vgcreate vgdata /dev/sdb1
sudo lvcreate -L 20G -n lvapp vgdata
```

20장. LAMP·Nginx·WAS·PHP-FPM: HTTP 요청이 코드 실행으로 이어지는 경로

> 브라우저가 server의 443 port에 연결했다고 application 코드가 바로 실행되는 것은 아니다. front web server가 TLS와 HTTP를 처리하고, static file은 직접 반환하며, dynamic request는 application runtime이나 WAS로 넘기고, application은 DB와 storage를 호출한다.

LAMP·Web Server·WAS

필기 확인 + 보충 해설

한 HTTP 요청이 여러 process를 거쳐 응답이 되는 구조

함께 찾을 말: LAMP, Apache, Nginx, WAS, Web Application Server, Server-side Script, JSP, Tomcat, IIS

LAMP·Web Server·WAS란 LAMP는 Linux·Apache·DB·server-side language의 전통 web stack이고, web server는 HTTP와 static/proxy를, WAS/application runtime은 dynamic code와 business logic을 처리한다.

LAMP stack

Linux, Apache, MySQL/MariaDB, PHP/Perl/Python의 조합을 가리키는 전통 용어다. 각 구성요소는 OS, web server, database, server-side language/runtime 역할을 나눈다. ELK stack처럼 stack은 함께 사용되는 기술 묶음을 뜻한다.

Web server와 WAS

web server는 HTTP connection, static content, TLS termination, reverse proxy를 담당한다. WAS는 application code 실행, routing, session, connection pool, business logic을 담당하는 server software를 뜻한다. 제품과 architecture에 따라 경계가 겹친다. Apache·Nginx가 module로 dynamic code를 실행하기도 하고, Tomcat·Gunicorn이 HTTP를 직접 받을 수도 있다.

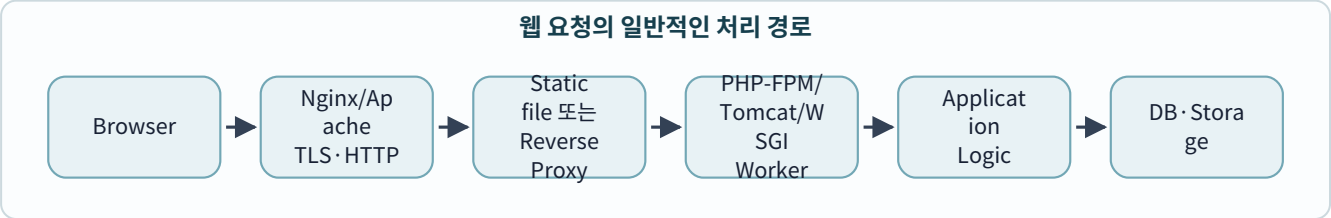


그림 20-1. 각 hop마다 timeout, body size, header, 권한, logging 설정이 있어 장애 원인을 층별로 확인한다.

Server-side script

browser가 받은 JavaScript가 client-side에서 실행되는 것과 달리 PHP, JSP, Python application code는 server에서 실행되어 HTML/JSON을 생성한다. client는 source code가 아니라 결과 response를 받는다. server-side script 자체와 이를 실행·관리하는 runtime/WAS를 구분한다.

CGI·FastCGI·PHP-FPM·WSGI·Worker

필기 확인 + 보충 해설

웹 서버가 직접 이해하지 못하는 언어 코드를 실행시키는 경계

함께 찾을 말: CGI, FastCGI, PHP-FPM, WSGI, ASGI, Worker, WAS Tuning

CGI·FastCGI·PHP-FPM·WSGI·Worker란 CGI 계열과 WSGI는 web server가 application code를 호출하는 interface이고, PHP-FPM과 application server는 지속 worker pool로 request를 실행·관리한다.

CGI

CGI는 web server가 request 정보를 environment variable과 stdin으로 external program에 전달하고 그 program의 stdout을 HTTP response로 사용하는 표준 interface다. request마다 process를 새로 만들면 overhead가 크다. FastCGI는 process를 지속 실행해 여러 request를 처리한다.

PHP-FPM

PHP FastCGI Process Manager는 PHP worker pool을 관리하고 Nginx/Apache에서 FastCGI request를 받아 PHP script를 실행한다. web server와 PHP runtime을 별도 process·user·socket으로 분리해 scaling과 privilege를 관리한다.

WSGI

WSGI는 Python web server와 Python web application/framework 사이의 callable interface 표준이다. Gunicorn, uWSGI 등의 server가 socket·worker를 관리하고 Flask/Django application callable을 호출한다. CGI처럼 매 request마다 새 process를 만들 필요가 없다. asynchronous application에는 ASGI가 널리 쓰인다.

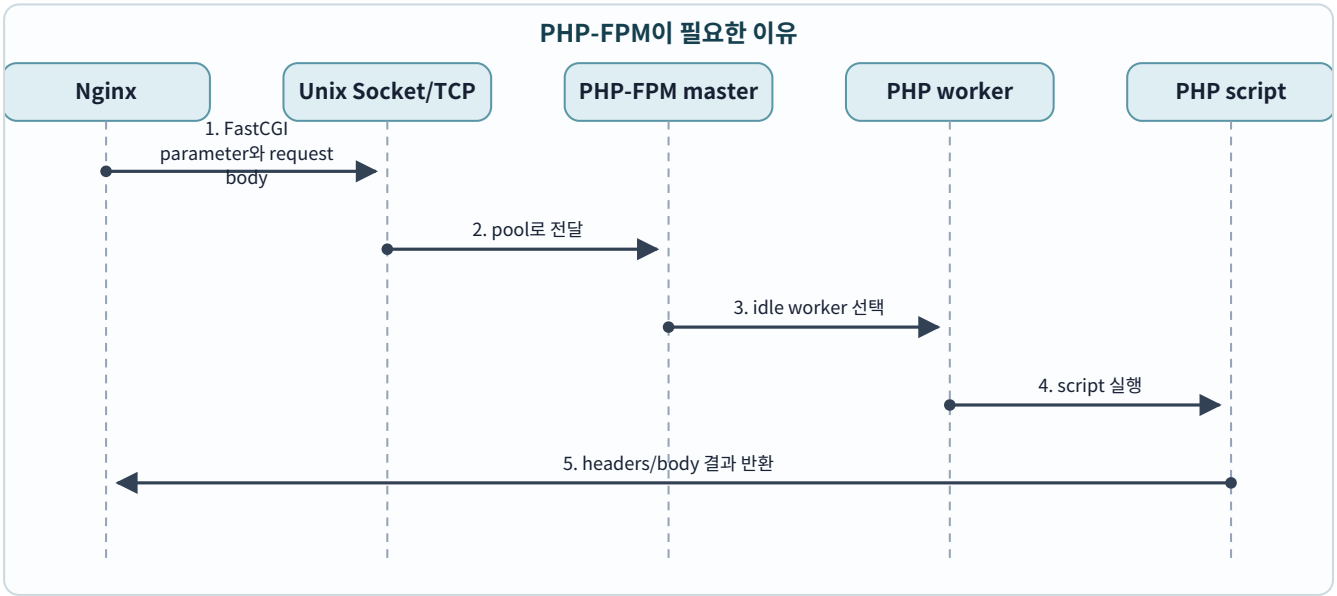


그림 20-2. Nginx는 PHP를 해석하지 않고 FastCGI로 PHP-FPM에 실행을 요청한다.

Worker

worker는 request나 task를 실제 처리하는 process/thread/event loop 실행 단위다. worker 수가 너무 적으면 queue가 길어지고, 너무 많으면 memory와 context switching, DB connection이 과도해진다. WAS tuning은 CPU-bound/I/O-bound 특성, request latency, memory, connection pool을 함께 본다.

Virtual Hosting·SNI·Sandboxing

필기 확인 + 보충 해설

한 서버에 여러 site를 안전하게 분리하는 구성

함께 찾을 말: IP-based Virtual Hosting, Port-based Virtual Hosting, Name-based Virtual Hosting, Host Header, SNI, conf.d, Sandbox, chroot, system user

Virtual Hosting·SNI·Sandboxing란 virtual hosting은 IP·port·hostname으로 여러 site를 구분하고, SNI는 TLS handshake에서 hostname을 알리며, sandboxing은 application process가 접근 가능한 자원 범위를 제한한다.

Virtual Hosting

한 server process/IP에서 여러 site를 구분해 hosting하는 기능이다. IP-based는 destination IP, port-based는 port, name-based는 HTTP Host header와 TLS SNI를 사용한다. 현대 web은 name-based가 일반적이다.

TLS와 SNI

HTTPS에서는 HTTP Host header보다 TLS handshake가 먼저이므로 server가 어느 certificate를 줄지 SNI(Server Name Indication)의 hostname을 사용한다. client가 SNI를 보내지 않으면 default virtual host certificate가 선택될 수 있다.

conf와 conf.d

주 설정 파일이 전체 구조와 include를 정의하고, `conf.d` directory의 여러 작은 파일을 포함해 site·module별 설정을 나누는 관례다. 이름 자체가 kernel 기능은 아니며 application이 include하도록 설정해야 한다.

Name-based virtual host와 PHP-FPM 연결

```
# Nginx 개념 예시
server {
    listen 80;
    server_name app.example.com;
    root /srv/app/public;

    location ~ \.php$ {
        include fastcgi_params;
        fastcgi_pass unix:/run/php/php-fpm.sock;
    }
}

nginx -t && systemctl reload nginx
```

Sandboxing과 service user

application을 전용 non-login user로 실행하고 filesystem write path, network destination, capability를 제한한다. systemd sandbox option, chroot/container, SELinux/AppArmor를 조합할 수 있다. chroot는 path view를 바꾸지만 완전한 security boundary로 단독 신뢰하기 어렵다.

WAS에 SSH를 열면 왜 위험한가

application 관리가 필요한 경우에도 shell 계정은 code·secret·DB credential·log에 직접 접근할 수 있는 강한 권한이다. 관리망, bastion, MFA, least privilege, file deployment pipeline으로 범위를 줄여야 한다.

21장. DB·DBMS·SQL·Hash: 웹 애플리케이션의 상태 저장

> 파일에도 데이터를 저장할 수 있지만 여러 사용자가 동시에 검색·수정하고 관계·무결성·복구를 관리하려면 DBMS가 필요하다. 이 장은 DB와 DBMS를 구분하고 account host, cursor, transaction, password hash를 연결한다.

DB·DBMS·RDBMS·SQL·Cursor

필기 확인 + 보충 해설

데이터 자체와 데이터를 관리하는 엔진을 구분하기

함께 찾을 말: Database, DBMS, RDBMS, SQL, NoSQL, Cursor

DB·DBMS·RDBMS·SQL·Cursor란 database는 데이터 집합, DBMS는 이를 transaction·동시성·권한과 함께 관리하는 software, RDBMS는 relational model을 사용하는 DBMS이며, cursor는 result를 순차 탐색하는 상태 객체다.

DB와 DBMS

database는 구조화된 데이터의 논리적 집합이고, DBMS는 이를 생성·조회·동시성·권한·복구·transaction으로 관리하는 software다. RDBMS는 table의 row/column과 relation, key, SQL을 중심으로 하는 DBMS 계열이다.

SQL과 NoSQL

SQL은 relational data definition·query·update·access control을 표현하는 언어다. NoSQL은 “SQL을 전혀 쓰지 않는다”보다 document, key-value, column-family, graph처럼 relational table 외의 data model과 scaling 특성을 가진 DB 계열을 넓게 부른다. 목적에 따라 함께 쓸 수 있다.

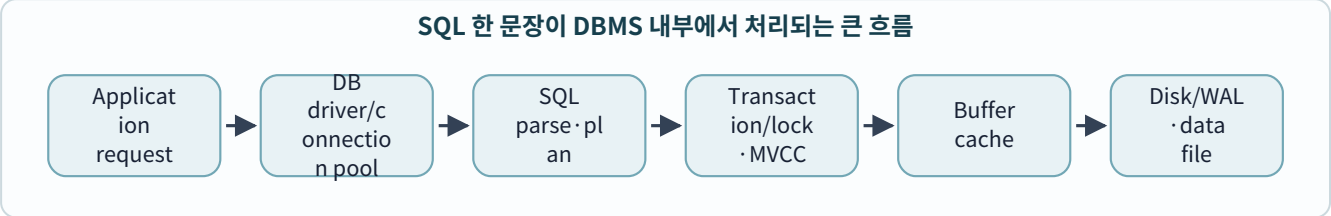


그림 21-1. SELECT 한 번도 network, parser, optimizer, cache, storage 층을 거친다.

Cursor

cursor는 query result의 현재 위치와 fetch 상태를 관리하는 handle이다. client library cursor는 SQL 실행과 row fetch interface를 제공하고, server-side cursor는 큰 result를 한꺼번에 memory에 올리지 않고 순차 처리할 수 있다. file cursor/offset도 “현재 읽을 위치”라는 공통 의미다.

DB 계정·Host·Bind Address

필기 확인 + 보충 해설

“DB 포트가 열림”과 “로그인·권한이 허용됨”은 다른 단계

함께 찾을 말: user@host, bind-address, MariaDB, MySQL, phpMyAdmin, Privilege

DB 계정·Host·Bind Address란 DB account는 username과 접속 origin 조건이 함께 식별되고, bind address는 DB server가 listen할 local interface를 제한하며, privilege는 account가 수행할 수 있는 DB 작업 범위를 정한다.

Account name과 접속 host가 한 쌍인 이유

MariaDB/MySQL 권한 모델에서는 계정을 `user@host`로 식별한다. `app@localhost`와 `app@10.%`는 이름이 같아도 서로 다른 account entry이며 인증 정보와 권한을 다르게 가질 수 있다. 어떤 항목이 match되는지 specificity와 server 규칙을 확인해야 한다.

bind-address

DB server가 어느 local IP interface에서 listen할지 정한다. 127.0.0.1이면 local connection 중심, 0.0.0.0이면 모든 IPv4 interface에 bind할 수 있다. listen 범위를 넓히는 것은 firewall·DB account host·TLS·network segmentation과 함께 판단해야 한다.

Application 계정

web application은 root DB account를 쓰지 않고 필요한 schema의 SELECT/INSERT/UPDATE 등 최소 privilege만 가진 전용 account를 사용한다. schema migration account와 runtime account를 분리하면 공격 영향 범위를 줄인다.

DB account와 network 범위

```
-- 개념 예시: 실제 비밀번호는 secret manager 사용
CREATE USER 'app'@'10.20.%' IDENTIFIED BY 'REDACTED';
GRANT SELECT, INSERT, UPDATE, DELETE ON appdb.* TO 'app'@'10.20.%';
SHOW GRANTS FOR 'app'@'10.20.%';

-- listen 상태는 OS에서도 확인
ss -lntp | grep 3306
```

phpMyAdmin

DB를 web UI로 관리하는 도구는 편리하지만 DB 권한을 browser에 노출하는 고가치 관리 surface다. Internet 공개를 피하고 관리망, MFA/reverse proxy, 최신 patch, 접근 로그를 적용한다.

Hash·Salt·Rainbow Table

필기 확인 + 보충 해설

비밀번호를 “암호화해서 저장”한다는 표현 바로잡기

함께 찾을 말: Hash Function, Password Hashing, Salt, Rainbow Table, Argon2, bcrypt, PBKDF2

Hash·Salt·Rainbow Table란 cryptographic hash는 입력을 고정 길이 digest로 바꾸는 단방향 함수이고, salt와 느린 password KDF는 동일 password 패턴과 사전 계산 공격을 어렵게 한다.

Hash function

임의 길이 입력을 고정 길이 digest로 바꾸는 함수다. cryptographic hash는 preimage resistance, second-preimage resistance, collision resistance 같은 특성을 목표로 한다. encryption처럼 key로 원문을 복호화하는 기능이 아니다.

Password hashing

password는 빠른 일반 hash 하나가 아니라 salt와 많은 계산·memory 비용을 사용하는 password hashing/KDF(Argon2, scrypt, bcrypt, PBKDF2 등)로 저장한다. login 시 같은 algorithm으로 candidate를 검증한다.

Rainbow table과 Salt

rainbow table은 가능한 password와 hash chain을 미리 계산해 빠른 hash 역추적을 줄이는 time-memory tradeoff 공격이다. 사용자별 random salt는 같은 password도 서로 다른 hash가 되게 해 precomputation을 재사용하기 어렵게 한다. salt는 secret일 필요는 없지만 고유하고 충분히 random해야 한다.

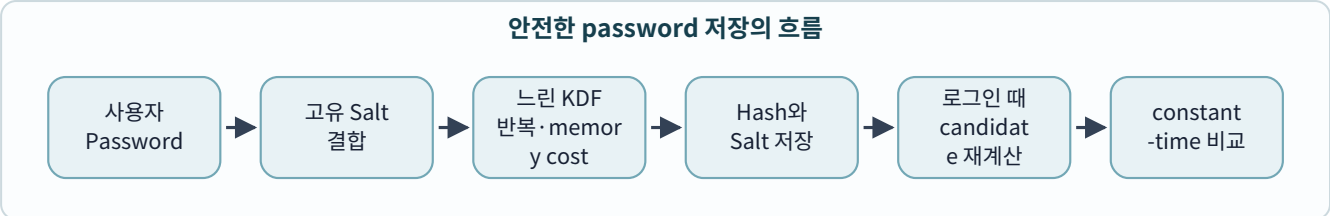


그림 21-2. DB가 유출되어도 공격자가 각 password guess에 큰 비용을 지불하게 만드는 것이 목표다.

Hash는 복호화하지 않는다

password 확인은 저장 hash를 “복호화”하는 것이 아니라 입력한 candidate로 hash/KDF를 다시 계산해 비교한다. 공격자는 추측 후보를 대량 시험한다.

CMS와 DB 운영

편의 기능이 많을수록 patch와 plugin 신뢰 관리가 중요하다

함께 찾을 말: CMS, WordPress, Joomla, Binary Log, Point-in-time Recovery

필기 확인 + 보충 해설

CMS와 DB 운영란 CMS는 content와 user를 관리하는 web application이고, DB backup·transaction log·최소 privilege는 application 장애와 침해에서 data를 복구·보호하는 운영 요소다.

CMS

Content Management System은 page·post·user·theme·plugin을 web UI에서 관리하는 application이다. WordPress, Joomla가 대표적이다. 개발 속도를 높이지만 core, plugin, theme, credential의 공급망·patch 관리가 공격면이 된다.

Application과 DB의 trust boundary

browser input은 application validation과 authorization을 거쳐 parameterized query로 DB에 전달해야 한다. DB account privilege도 최소화해 SQL injection이 발생해도 접근 범위를 줄인다.

Backup과 binary log

DB backup은 consistent snapshot/dump와 transaction log를 함께 관리해야 point-in-time recovery가 가능하다. MySQL/MariaDB binary log는 data change event를 기록해 replication과 복구에 쓰지만 credential·personal data가 포함될 수 있어 보호와 retention이 필요하다.

22장. DNS·FTP·SMB·NFS·CDN: 이름과 파일을 network service로 제공하기

DNS는 domain 이름을 단순히 IP 하나로 바꾸는 전화번호부가 아니다. 계층적 namespace, zone delegation, 여러 record type, cache와 TTL을 이용하는 분산 database다. 이 장은 BIND 설정에서 보이는 @, FQDN, zone, reverse lookup을 설명하고 file service를 연결한다.

DNS·Domain·FQDN·Zone

필기 확인 + 보충 해설

이름을 IP 하나로 바꾸는 것보다 넓은 분산 데이터베이스

함께 찾을 말: Domain Name System, Domain, Root DNS, TLD, FQDN, Zone, Authoritative Server, Recursive Resolver

DNS·Domain·FQDN·Zone란 DNS는 계층적 domain name과 다양한 resource record를 분산 zone으로 관리하고 cache하는 naming system이며, FQDN은 root까지 완전히 지정한 이름, zone은 특정 server가 authoritative하게 관리하는 namespace 범위다.

Domain tree와 root

DNS 이름은 오른쪽에서 왼쪽으로 root(.), top-level domain, 하위 label의 계층을 이룬다. `www.example.com.`처럼 끝의 dot까지 쓴 이름은 root까지 포함한 absolute/FQDN이다. 일반 입력에서 마지막 dot를 생략하면 resolver search domain이 붙을 수 있다.

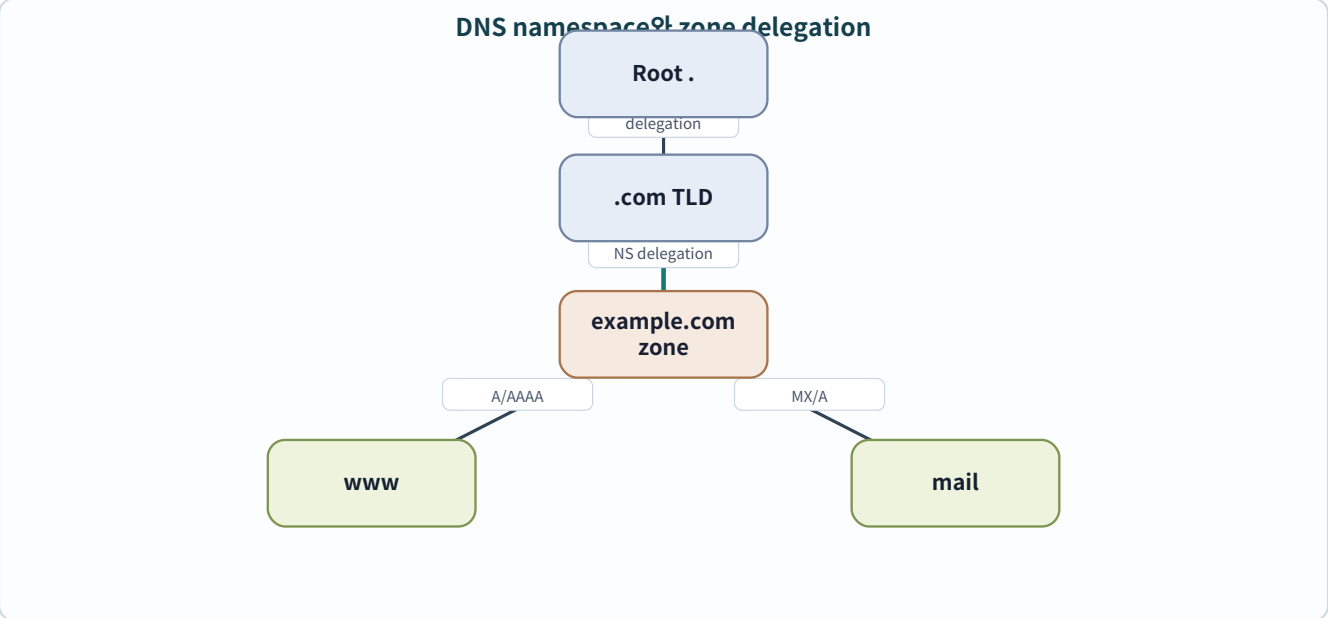


그림 22-1. parent zone은 child의 authoritative server를 NS record로 위임한다.

FQDN과 absolute path 비유

FQDN은 root부터 완전히 지정한 DNS 이름이고, absolute filesystem path는 /부터 완전히 지정한 경로다. 둘 다 현재 context/search path에 의존하지 않는다는 공통점이 있지만 서로 다른 namespace다.

Resolver와 authoritative server

stub resolver는 OS/library에서 recursive resolver에 질문한다. recursive resolver는 cache를 확인하고 root·TLD·authoritative server를 따라가 답을 얻는다. authoritative server는 자신이 책임지는 zone data를 근거로 답한다.

DNS Record·TTL·정방향/역방향 조회

필기 확인 + 보충 해설

record type별 의미와 cache 동작

함께 찾을 말: A Record, AAAA, CNAME, MX, NS, SOA, PTR, TXT, SRV, DNS TTL, Forward DNS, Reverse DNS, @

DNS Record·TTL·정방향/역방향 조회란 DNS resource record는 이름에 address·mail·name server 등의 typed data를 연결하고, TTL은 cache 유효 시간을, reverse DNS는 address namespace에서 PTR name을 조회하는 방식을 뜻한다.

대표 record

Record	의미
A	이름의 IPv4 address
AAAA	이름의 IPv6 address
CNAME	한 이름을 다른 canonical name의 alias로 연결
MX	domain의 mail exchanger와 priority
NS	zone의 authoritative name server
SOA	zone 시작·serial·timer·관리 정보
PTR	reverse tree에서 address를 name으로 mapping
TXT	정책·검증 등 text data
SRV	service의 target host와 port·priority·weight

TTL

record를 resolver가 얼마나 오래 cache할 수 있는지 초 단위로 표시한다. TTL이 길면 authoritative query와 latency를 줄이지만 변경 반영이 늦고, 짧으면 빠른 전환이 가능하지만 query 부하가 늘어난다. 이미 cache된 record의 TTL은 변경 직전에 낮추어야 효과가 있다. IP packet TTL과 이름만 같고 의미는 다르다.

Zone file의 @와 relative name

BIND zone file에서 @는 현재 zone origin을 뜻한다. owner name 끝에 dot가 없으면 origin이 붙는다. SOA의 관리자 mailbox는 첫 dot가 email의 @를 대신하고 literal dot는 escape하는 zone file 표기 규칙이다.

Forward와 Reverse DNS

forward lookup은 name→address, reverse lookup은 address를 in-addr.arpa(IPv4)나 ip6.arpa(IPv6) 이름으로 뒤집어 PTR record를 찾는다. reverse record가 있다고 forward와 자동 일치하는 것은 아니며 각 zone 관리자가 별도 설정한다.

BIND zone file 개념 예시

```
$ORIGIN example.com.
$TTL 300
@ IN SOA ns1.example.com. admin.example.com. (
  2026072601 3600 900 1209600 300 )
@    IN NS   ns1.example.com.
ns1  IN A    192.0.2.53
www  IN A    192.0.2.80
@    IN MX 10 mail.example.com.
mail IN A    192.0.2.25
```

BIND·named·DNSSEC·dig

필기 확인 + 보충 해설

DNS 설정과 실제 응답을 분리해 검증하는 도구

함께 찾을 말: BIND, named, bind-utils, dig, nslookup, DNSSEC, ipconfig / flushdns

BIND·named·DNSSEC·dig란 BIND는 DNS server software, named는 그 daemon이고, dig는 DNS query를 세밀하게 확인하는 client tool이며, DNSSEC은 signed record chain으로 응답의 origin과 무결성을 검증한다.

BIND와 named

BIND는 널리 쓰이는 DNS software suite이고 named는 authoritative/recursive DNS daemon 실행 파일 이름이다. bind-utils/dnsutils에는 dig, nslookup, host 같은 client tool이 포함된다. distribution마다 package 이름과 설정 경로가 다르다.

Zone 구성의 큰 요소

options에서 listen address, recursion, allow-query, forwarder, DNSSEC validation 등을 설정하고, zone statement에서 type(primary/secondary 등), file, transfer policy를 지정한다. authoritative-only server에 recursion을 불필요하게 열면 amplification과 cache attack surface가 커진다.

DNS 진단과 설정 검사

```
# 조회
 dig www.example.com A +trace
 dig @192.0.2.53 example.com SOA
 dig -x 192.0.2.80
 nslookup www.example.com

# BIND 검증(배포판 경로에 맞춤)
 named-checkconf
 named-checkzone example.com /var/named/example.com.zone
```

Client cache flush

Windows의 `ipconfig /flushdns`는 local DNS client cache를 비운다. browser, application, recursive resolver, CDN이 별도 cache를 가질 수 있어 한 곳을 flush했다고 전체 Internet에 즉시 반영되는 것은 아니다.

DNSSEC

DNS record에 digital signature를 추가해 resolver가 data origin과 무결성을 검증하도록 한다. confidentiality를 제공하지 않으며 query 내용은 별도 DoT/DoH 같은 channel 보호가 필요하다. key rotation과 delegation chain을 정확히 운영해야 한다.

FTP·SFTP·SMB/CIFS·Samba·NFS

필기 확인 + 보충 해설

파일을 복사하는 protocol과 remote directory를 mount하는 protocol

함께 찾을 말: FTP, vsftpd, SFTP, FTPS, SMB, CIFS, Samba, NFS, mount.cifs

> **FTP·SFTP·SMB/CIFS·Samba·NFS란** FTP/SFTP는 file transfer session을 제공하고, SMB와 NFS는 remote file share를 filesystem처럼 접근하게 하며, Samba는 SMB를 Unix/Linux에서 구현한다.

FTP

FTP는 control connection과 별도 data connection을 사용한다. active mode에서는 server가 client로 data connection을 열고, passive mode에서는 client가 server가 알려준 port로 연결한다. NAT/firewall을 지날 때 data channel 범위 설정이 필요하다. 기본 FTP 인증과 data는 평문이므로 SFTP(SSH subsystem) 또는 FTPS(TLS)를 구분해 사용한다.

vsftpd와 chroot

vsftpd는 Linux FTP server daemon이다. local user, anonymous, write, passive port, chroot를 설정할 수 있다. chroot는 user의 filesystem view를 제한하지만 kernel privilege와 configuration 취약점을 모두 막는 sandbox는 아니다.

SMB/CIFS와 Samba

SMB는 Windows file/printer sharing protocol이고 CIFS는 오래된 SMB dialect를 가리키는 이름으로도 쓰인다. Samba는 Unix/Linux에서 SMB server/client와 domain integration을 구현한다. mount.cifs로 remote share를 filesystem에 연결할 수 있다.

NFS

NFS는 Unix 계열 network filesystem protocol로 server의 export를 client directory에 mount한다. user identity mapping, root squash, network trust, file locking, version을 설계해야 한다.

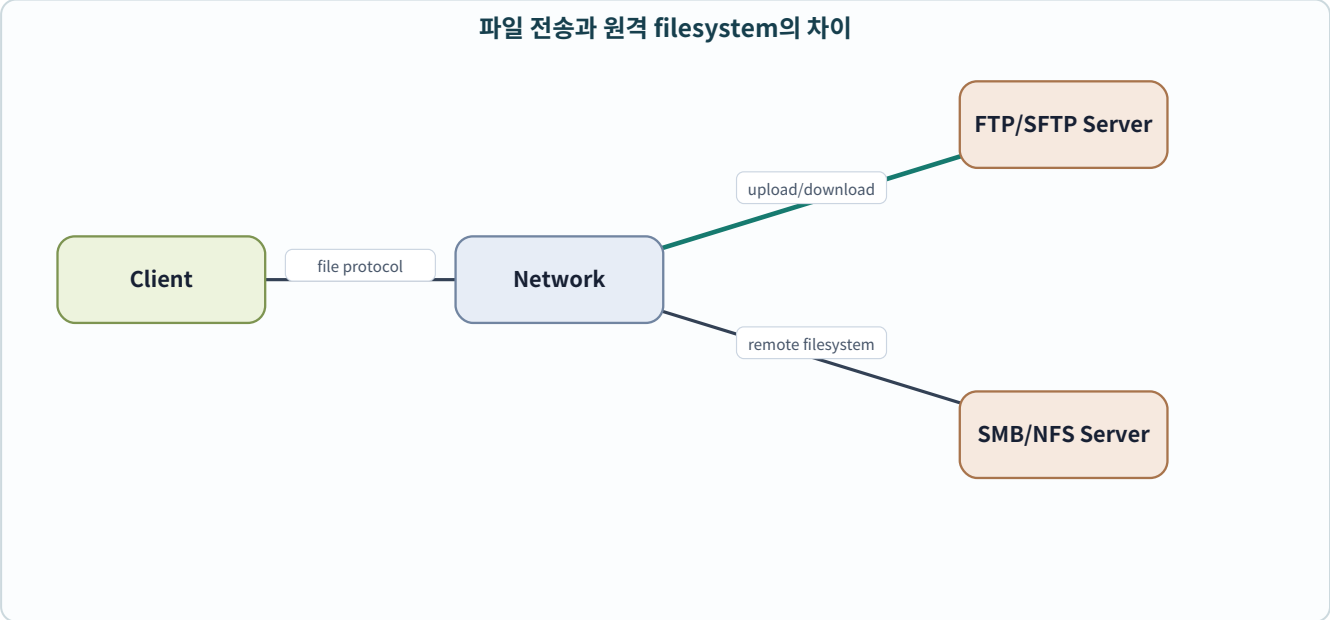


그림 22-2. FTP는 명시적 file transfer, SMB/NFS는 remote directory를 지속적으로 file namespace에 제공한다.

rsync·Mirror·CDN·Cache

필기 확인 + 보충 해설

복제·동기화·cache가 가용성과 최신성에 미치는 영향

함께 찾을 말: rsync, Mirror Domain, CDN, Cache

rsync·Mirror·CDN·Cache란 rsync는 file 차이를 계산해 효율적으로 동기화하고, mirror는 content 복제본, CDN은 분산 edge cache로 사용자에게 content를 제공하며, cache는 가까운 계층에 결과를 임시 저장한다.

rsync

rsync는 source와 destination의 file metadata와 block 차이를 비교해 변경분 중심으로 동기화한다. local, SSH remote shell, rsync daemon mode가 있다. 삭제 option은 destination의 불필요 file을 제거할 수 있어 dry-run과 path trailing slash 의미를 확인한다.

Mirror와 CDN

mirror는 동일 content를 다른 server/domain에 복제해 download 분산·가용성을 높인다. CDN은 edge location에 content를 cache하고 DNS/anycast/routing으로 client를 가까운 edge에 연결한다. dynamic acceleration, TLS, WAF를 함께 제공하기도 한다.

Cache

cache는 원본보다 가까운 빠른 계층에 최근 결과를 저장해 latency와 origin load를 줄인다. stale data, invalidation, cache key, private data leakage가 핵심 문제다. DNS TTL, browser cache, reverse proxy cache, DB buffer cache는 같은 아이디어를 다른 층에 적용한다.

동기화 전 dry-run

```
# source directory 내용만 destination으로 동기화
rsync -av --dry-run /srv/data/ backup:/srv/data/
rsync -av --delete /srv/data/ backup:/srv/data/  # 주의

# NFS/SMB mount 확인
findmnt -t nfs,nfs4,cifs
```

23장. Log·rsyslog·NTP·SNMP·RRD: 운영 상태를 증거로 남기기

> 장애와 침해가 끝난 뒤 “무슨 일이 있었는가”를 재구성하려면 시간, source, severity, event 내용이 일관되게 남아야 한다. 로그를 생성하는 것과 중앙으로 전송·보존·검색·경보하는 것은 서로 다른 단계다.

Log·rsyslog·Severity·Trap

필기 확인 + 보충 해설

기록 한 줄이 중앙 경보가 되기까지

함께 찾을 말: /var/log, tail, head, xferlog, rsyslogd, Syslog Severity, logging trap

> **Log·rsyslog·Severity·Trap란** log는 system event를 시간과 context와 함께 기록한 증거이고, rsyslog는 이를 filter·저장·전송하며, severity는 중요도를, trap은 비동기 event notification을 나타낸다.

/var/log와 daemon log

Linux의 전통적인 log 위치지만 systemd journal, container stdout, application 전용 directory 등 여러 경로가 있다. `tail -f`는 file 끝을 따라가고, head는 앞부분을 본다. xferlog는 FTP transfer log 형식·파일 이름으로 쓰였다.

rsyslog

syslog message를 local file, remote server, queue, database 등으로 routing하는 daemon이다. facility와 severity, program name, property filter를 사용한다. UDP 전송은 간단하지만 손실과 spoofing 위험이 있고, TCP/TLS와 disk queue를 통해 신뢰성을 높일 수 있다.

Cisco logging severity

번호	이름	의미
0	emergencies	system unusable
1	alerts	즉시 조치 필요
2	critical	critical condition
3	errors	error
4	warnings	warning
5	notifications	정상이나 중요한 사건
6	informational	정보
7	debugging	상세 debug

Trap

trap은 정상 흐름을 가로채는 event라는 넓은 뜻이 있고, SNMP trap은 agent가 manager의 polling을 기다리지 않고 event를 비동기 notification으로 보내는 message다. Cisco `logging trap`은 remote syslog로 보낼 최소 severity 수준을 설정하는 문맥이다.



그림 23-1. 각 단계에서 time zone, hostname, field parsing, queue, retention이 맞아야 검색 가능한 증거가 된다.

Logrotate·NTP·Serialization

필기 확인 + 보충 해설

증거의 용량·시간·형식을 일관되게 유지한다

함께 찾을 말: Log Rotation, logrotate, NTP, Time Synchronization, Serialization, Binary Logging

Logrotate·NTP·Serialization란 logrotate는 log file의 크기와 보존을 관리하고, NTP는 system time을 동기화하며, serialization은 data structure를 저장·전송 가능한 형식으로 변환한다.

Log rotation

지속 증가하는 log file을 일정 크기·주기마다 rename/compress하고 오래된 file을 삭제해 disk 고갈을 막는다. application이 열린 file descriptor를 계속 쓰면 rename 후 옛 inode에 기록할 수 있어 signal로 reopen하거나 copytruncate를 쓰는데, copytruncate는 짧은 유실·중복 가능성이 있다.

NTP

Network Time Protocol은 여러 time source와 offset·delay를 측정해 system clock을 점진적으로 동기화한다. log correlation, certificate validity, Kerberos, distributed transaction에서 시간이 맞아야 한다. 단순히 매번 시계를 강제로 점프시키면 application timer에 영향을 줄 수 있어 slew와 step 정책을 사용한다.

Serialization

memory의 객체·구조를 file이나 network로 전송 가능한 byte/text 형식으로 바꾸는 과정이다. JSON, Protocol Buffers, Python pickle 등이 있다. 신뢰하지 않는 pickle/Java serialization을 역직렬화하면 code execution이 가능한 형식도 있어 data format의 안전 속성을 구분한다.

로그 회전과 시간 동기화 확인

```
# logrotate 설정 확인
logrotate -d /etc/logrotate.conf
journalctl --since '1 hour ago'

# 시간 상태
timedatectl
chronyc tracking
chronyc sources -v
```

Log injection

사용자 입력의 줄바꿈·제어문자를 그대로 log에 넣으면 가짜 event line을 만들거나 viewer에서 script가 실행될 수 있다. structured logging, escaping, field length 제한과 viewer output encoding이 필요하다.

SNMP·OID·MIB·RRD

필기 확인 + 보충 해설

장비 counter가 장기 trend graph로 바뀌는 과정

함께 찾을 말: SNMP, SNMP Manager, SNMP Agent, OID, MIB, Community String, SNMPv1, SNMPv2c, SNMPv3, RRD, RRA, RRDTool, MRTG, Cacti

SNMP·OID·MIB·RRD란 SNMP는 manager와 agent가 OID로 device 상태를 조회·설정하고 event를 알리는 관리 protocol이며, MIB는 OID schema, RRD는 고정 용량 time-series 저장 구조다.

Manager와 Agent

SNMP manager가 network device의 agent에 OID 값을 GET/GETNEXT/GETBULK로 요청하고 SET으로 변경할 수 있으며, agent는 trap/inform을 보낼 수 있다. 실제 운영은 read-only와 management network 제한을 기본으로 한다.

OID와 MIB

OID는 점으로 이어진 숫자 tree path로 관리 object를 식별하고, MIB는 OID의 이름, type, access, 의미를 설명하는 schema 문서다. MIB file이 device data 자체는 아니다.

Community String과 Version

SNMPv1/v2c의 community string은 사실상 shared password처럼 request에 포함되지만 암호화되지 않아 packet capture에 노출될 수 있다. SNMPv3는 user-based authentication과 privacy encryption, access control을 제공한다.

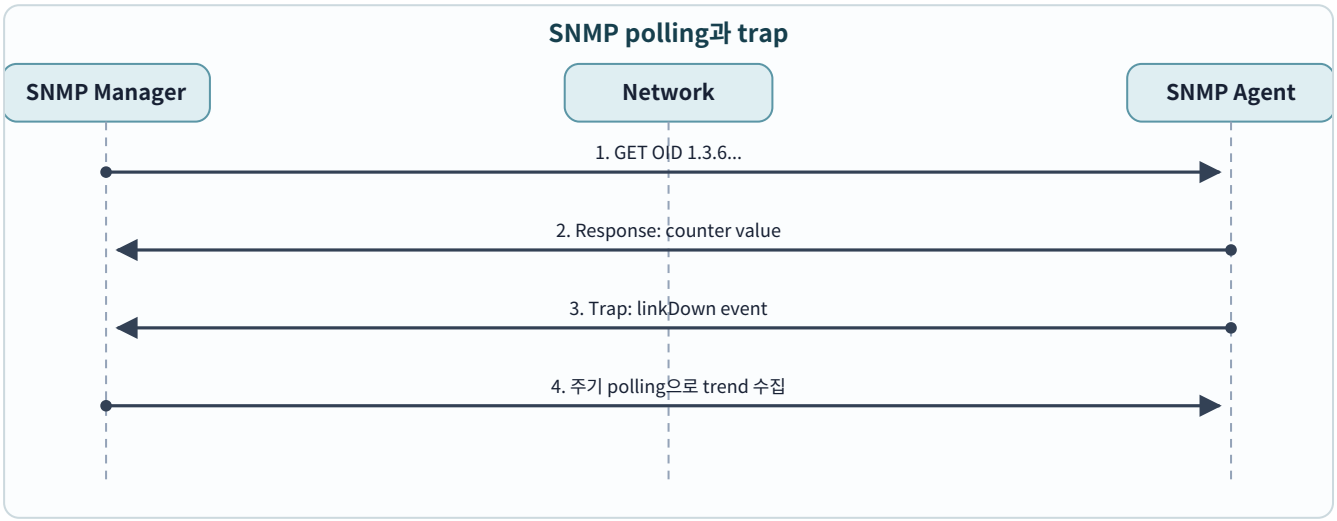


그림 23-2. counter는 두 시점의 차이를 시간으로 나눠 rate를 계산하는 경우가 많다.

RRD·RRA·MRTG·Cacti

RRDTool은 고정 크기 round-robin database에 time-series를 저장하며 시간이 지나면 오래된 data를 평균·최대값 등으로 consolidation한다. RRA는 보존 해상도와 기간을 정의한다. MRTG는 network traffic graph를 만들고, Cacti는 SNMP polling, template, RRD graph를 web UI로 관리한다.

PART V

보안 통제·패킷 경로·탐지·관제

보안 제품 이름을 외우기 전에 무엇을 보호하고, 어느 지점에서 어떤 evidence를 관찰하며, 허용·차단·탐지를 누가 결정하는지 이해해야 한다. 이 부분은 암호·접근 통제부터 Linux kernel packet path, IDS/IPS, SIEM과 대응까지 하나의 방어 흐름으로 묶는다.

24장. 보안의 기본 언어: 자산·위협·취약점·위험·접근통제

> 보안 도구는 “나쁜 traffic을 막는 프로그램”이 아니라 특정 자산과 신뢰 경계에 통제를 배치하는 수단이다. 무엇이 잘못될 수 있는지 말하려면 threat, vulnerability, risk, exploit, control을 구분해야 한다.

자산·위협·취약점·위험

필기 확인 + 보충 해설

도구 선택보다 먼저 보호 문제를 정확히 서술한다

함께 찾을 말: Asset, Threat, Vulnerability, Exploit, Impact, Risk, Attack Surface, Trust Boundary, Defense in Depth

자산·위협·취약점·위험란 자산은 보호 대상, 위협은 해를 줄 가능성이 있는 조건, 취약점은 악용 가능한 약점, 위험은 발생 가능성과 영향을 고려한 손실 가능성이다.

자산·위협·취약점

Asset

보호 가치가 있는 data, service, account, key, 장비, reputation.

Threat

자산에 해를 줄 수 있는 actor·event·condition. 공격자, 화재, 실수, 공급망 장애도 포함.

Vulnerability

위협이 악용할 수 있는 설계·구현·설정·운영의 약점.

Exploit

취약점을 실제로 이용해 의도하지 않은 동작을 일으키는 방법·code.

Impact

사건이 발생했을 때 business·기술적으로 생기는 손실.

Risk

가능성과 impact를 고려해 우선순위를 정한 불확실한 손실의 정도.



그림 24-1. 취약점 severity가 높아도 외부 노출·자산 가치·보완 통제에 따라 조직 risk는 달라진다.

Attack surface와 trust boundary

attack surface는 attacker가 접근·입력·상호작용할 수 있는 endpoint, port, API, account, dependency의 총합이다. trust boundary는 data나 identity를 다른 신뢰 수준의 component로 넘길 때 validation·authentication·authorization이 필요한 경계다.

Defense in depth

한 통제가 실패해도 다른 독립 통제가 피해를 제한하도록 network segmentation, MFA, least privilege, application validation, EDR, backup을 겹치는 전략이다. 같은 password를 여러 제품에 쓰는 것처럼 공통 실패 원인을 공유하면 층이 많아도 독립적 방어가 아니다.

CIA·인증·인가·최소 권한

필기 확인 + 보충 해설

로그인과 권한을 같은 기능으로 보지 않기

함께 찾을 말: CIA Triad, Confidentiality, Integrity, Availability, Authentication, Authorization, Accounting, Least Privilege, Default Deny

CIA·인증·인가·최소 권한란 CIA는 기밀성·무결성·가용성의 보안 목표이고, 인증은 identity를 확인하며, 인가는 identity의 action 권한을 결정하고, 최소 권한은 필요한 범위만 허용한다.

CIA

Confidentiality

허가된 주체만 정보를 읽을 수 있게 한다. encryption, ACL, data masking.

Integrity

정보와 system 상태가 허가 없이 바뀌지 않았음을 보장한다. hash/MAC/signature, change control.

Availability

필요할 때 service와 data를 사용할 수 있게 한다. redundancy, rate limit, backup, DDoS protection.

Authentication과 Authorization

authentication은 “누구인가”를 검증하고, authorization은 그 identity가 “무엇을 할 수 있는가”를 결정한다. session은 authentication 결과를 여러 request에 유지한다. accounting/auditing은 무엇을 했는지 기록한다.

Least privilege와 default deny

필요한 작업에 필요한 최소 권한만 부여하고, 명시적으로 허용하지 않은 접근은 거부한다. 관리자 편의를 위해 broad permission을 먼저 주고 나중에 줄이는 방식은 숨은 의존성과 공격면을 만든다.

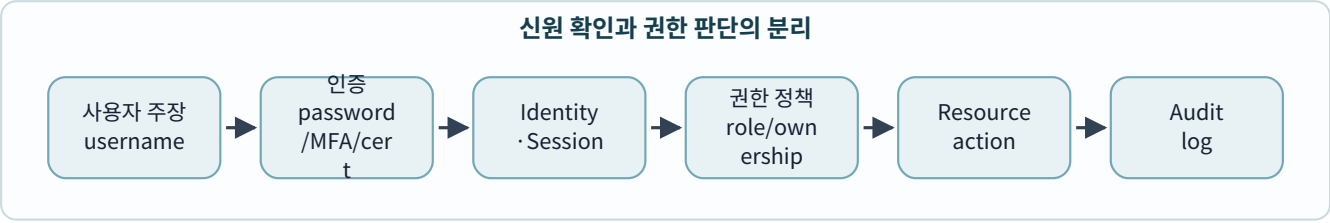


그림 24-2. 로그인 성공은 모든 resource 접근 허가가 아니다. 매 요청에서 object와 action 권한을 검사한다.

대칭키·비대칭키·Hash·Digital Signature

필기 확인 + 보충 해설

암호화라는 한 단어 안의 서로 다른 목적

함께 찾을 말: Symmetric Cryptography, Asymmetric Cryptography, AES, RSA, ECC, Hash, Digital Signature, AEAD, Key Size

대칭키·비대칭키·Hash·Digital Signature란 대칭키 암호는 shared secret으로 data를 암복호화하고, 비대칭키는 public/private key로 key agreement·암호화·서명을 수행하며, hash는 data digest, digital signature는 private key 기반 무결성·origin 검증을 제공한다.

대칭키 암호

같은 secret key로 encryption과 decryption을 수행한다. AES가 대표적이다. 대량 data에 효율적이지만 통신 당사자가 secret key를 안전하게 공유해야 한다. mode/nonce/IV를 잘못 쓰면 강한 algorithm도 안전하지 않다. AEAD(AES-GCM, ChaCha20-Poly1305)는 confidentiality와 integrity를 함께 제공한다.

비대칭키 암호

public key와 private key의 쌍을 사용한다. public key encryption, digital signature, key agreement 목적이 서로 다르며 RSA, ECDSA, Ed25519, ECDH 같은 algorithm을 용도에 맞게 쓴다. private key는 소유자가 보호하고 public key의 신뢰 연결은 certificate/PKI가 담당한다.

Key size를 단순 비교하면 안 되는 이유

AES-256의 256bit와 RSA-2048의 2048bit는 같은 security strength 척도가 아니다. 대칭·RSA·elliptic curve는 공격 수학과 key 구조가 다르므로 algorithm family와 권장 parameter를 표준 문서로 확인한다. version number도 암호 강도 bit 수와 별개다.

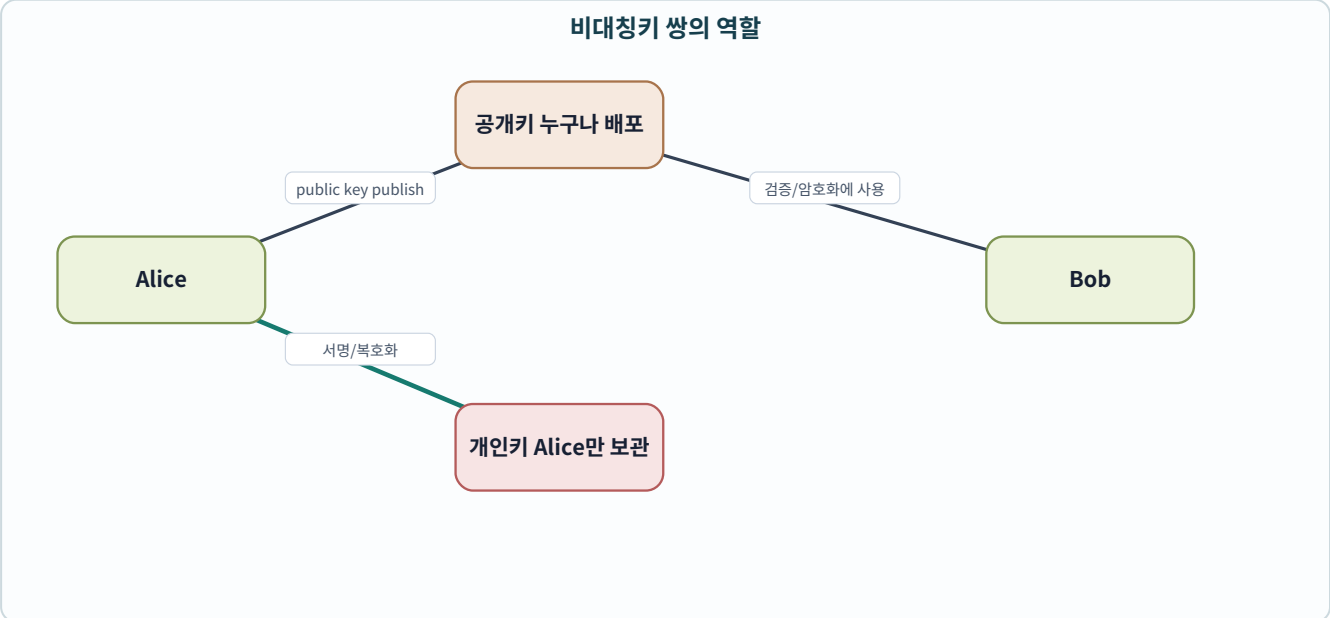


그림 24-3. 누가 어느 key로 무엇을 하는지는 encryption과 signature에서 반대처럼 보일 수 있어 목적을 먼저 말한다.

Hash, encryption, encoding

encoding은 표현 형식 변환으로 secret이 필요 없고 되돌릴 수 있다. encryption은 key로 confidentiality를 제공하고 복호화한다. hash는 고정 길이 digest로 변환하며 원문 복원을 목표로 하지 않는다.

TLS · Certificate · PKI · SSH

필기 확인 + 보충 해설

평문 protocol을 단순히 “암호화 비트 수”로 설명하지 않기

함께 찾을 말: SSL, TLS, Certificate, PKI, CA, SNI, SSH, Host Key

TLS · Certificate · PKI · SSH란 TLS는 certificate 기반 상대 인증과 key agreement 후 application traffic을 암호화·무결성 보호하고, PKI는 public key와 identity의 신뢰 chain을 관리하며, SSH는 암호화된 원격 관리 channel을 제공한다.

TLS handshake의 목적

client와 server가 protocol version과 cipher suite를 협상하고, server certificate를 검증하며, ephemeral key agreement로 shared traffic key를 만들고, handshake message 무결성을 확인한다. 이후 application data는 대칭 AEAD로 보호한다.

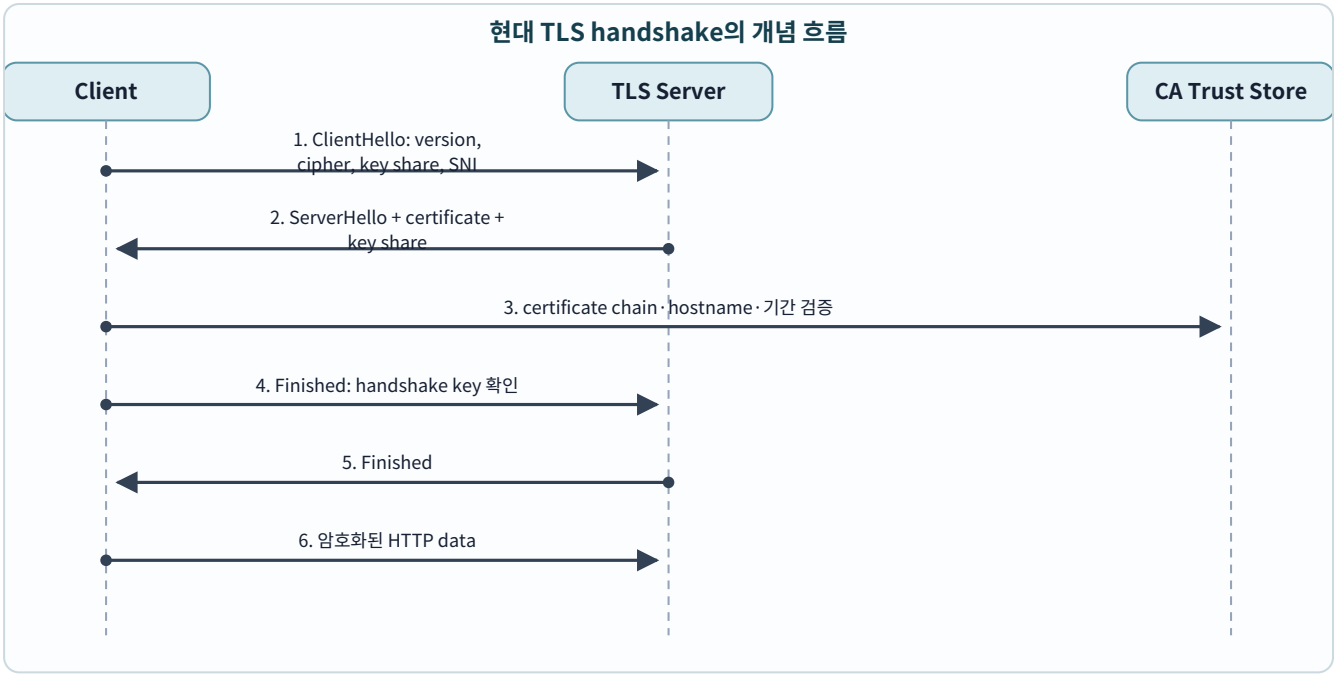


그림 24-4. 실제 TLS 1.3 message와 optional client authentication은 더 세부적이지만 핵심은 identity와 session key 합의다.

Certificate와 CA

certificate는 subject name, public key, validity, issuer, extension을 서명된 구조로 묶는다. client는 trusted root까지 chain, hostname/SAN, 기간, key usage, revocation 정책을 검증한다. certificate가 암호화를 “담고” 있는 것이 아니라 public key와 identity binding을 제공한다.

SSH

SSH는 server host key로 상대를 식별하고 key exchange로 session key를 만든 뒤 password/public-key authentication과 encrypted shell, file transfer, port forwarding을 제공한다. 처음 접속한 host key를 무심코 accept하면 MITM을 놓칠 수 있다.

SSL과 TLS

SSL은 구형 protocol 이름이며 SSLv2/v3는 안전하지 않아 사용하지 않는다. 현대 configuration은 TLS 1.2/1.3과 안전한 cipher, certificate 관리가 중심이다.

25장. Linux Host 보안: sudo·SELinux·서비스 격리·Fail2ban

> Linux permission만으로 모든 접근을 막을 수 없다. root 권한 위임, process domain, network exposure, service user, authentication log를 함께 설계해야 한다.

sudo·Linux 계정·Service User

필기 확인 + 보충 해설

root를 모두에게 나누는 대신 작업 단위 권한을 위임한다

함께 찾을 말: sudo, /etc/passwd, /etc/shadow, /sbin/nologin, System User, System Group

sudo·Linux 계정·Service User란 sudo는 policy 기반으로 다른 identity의 command 실행을 위임하고, Linux account는 UID/GID·home·shell로 process와 file ownership을 구분하며, service user는 daemon 권한을 격리한다.

sudo

sudo는 허가된 user가 policy에 따라 다른 user, 흔히 root 권한으로 특정 command를 실행하게 한다. root password를 공유하지 않고 개인 identity와 log를 유지할 수 있다. `sudo su -`처럼 광범위 shell을 열기보다 command와 argument를 최소화한다.

/etc/passwd와 /etc/shadow

/etc/passwd는 account name, UID, GID, comment, home, login shell field를 저장하며 password hash는 보통 x placeholder이고 실제 hash는 root만 읽을 수 있는 /etc/shadow에 있다. `/sbin/nologin`은 interactive login을 거부하는 shell로 service account에 사용된다.

passwd field	의미
name	account 이름
x	shadow 사용 표시
UID	user number
GID	primary group
GECOS	설명
home	home directory
shell	login program

Service account와 system group

daemon별 UID/GID를 분리하면 file permission, process ownership, audit를 통해 침해 영향 범위를 제한한다. 동일 www-data user로 모든 site를 실행하면 한 site의 file read 취약점이 다른 site secret으로 번질 수 있다.

identity와 sudo 정책 확인

```
id nginx
getent passwd nginx
sudo -l
sudo visudo -c
ps -eo user,pid,cmd | grep -E 'nginx|php-fpm'
```

SELinux

필기 확인 + 보충 해설

파일 소유자 권한을 넘어 process 역할을 제한한다

함께 찾을 말: Security-Enhanced Linux, MAC, Mandatory Access Control, Type Enforcement, AVC, Enforcing, Permissive

SELinux란 SELinux는 process와 object에 label을 붙이고 중앙 policy로 허용 관계를 강제하는 Linux Mandatory Access Control 시스템이다.

DAC와 MAC

일반 rwx/ACL은 object owner가 권한을 변경할 수 있는 Discretionary Access Control이다. SELinux는 system policy가 process domain과 object type 사이의 허용 관계를 강제하는 Mandatory Access Control이다. DAC에서 허용되어도 SELinux가 거부할 수 있다.

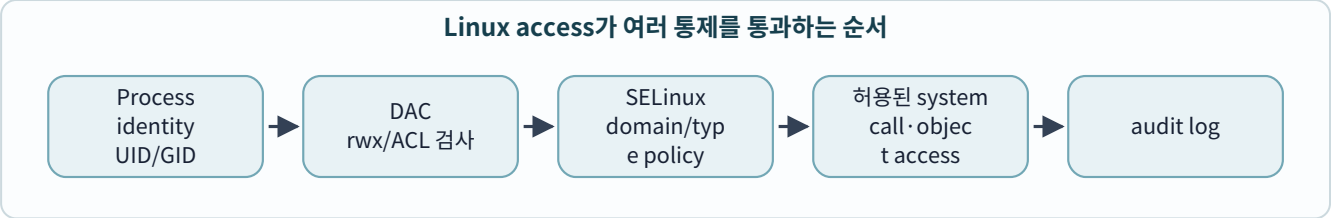


그림 25-1. root process라도 SELinux domain 정책에 따라 특정 file·port 접근이 거부될 수 있다.

Label과 Type Enforcement

file과 process에 security context(user:role:type:level)가 붙고 type enforcement rule이 domain이 어떤 type에 어떤 class permission을 가질지 정한다. web content는 httpd_sys_content_t, writable content는 별도 type이 필요한 식이다.

Enforcing·Permissive·Disabled

enforcing은 policy를 실제 차단하고 log, permissive는 차단하지 않고 would-deny를 log, disabled는 SELinux를 사용하지 않는다. 문제 해결을 위해 무조건 끄기보다 AVC log, file context, boolean, custom policy 순으로 원인을 찾는다.

SELinux 상태와 label 확인

```
getenforce
sestatus
ls -Z /var/www/html
ps -eZ | grep httpd
ausearch -m AVC -ts recent
restorecon -Rv /var/www/html
getsebool -a | grep httpd
```

chmod 777로 해결되지 않는 이유

DAC permission이 넓어져도 SELinux가 domain→type 접근을 거부하면 계속 실패한다. 반대로 SELinux가 허용해도 DAC가 거부하면 접근할 수 없다. 두 정책을 각각 확인한다.

Fail2ban·SSH Brute Force·TCP Wrapper

필기 확인 + 보충 해설

로그 기반 자동 차단이 인증 보안을 완전히 대체하지 않는다

함께 찾을 말: Fail2ban, SSH Brute Force, Jail, TCP Wrapper, hosts.allow, hosts.deny

Fail2ban·SSH Brute Force·TCP Wrapper란 Fail2ban은 log의 반복 실패를 탐지해 source를 일시 차단하고, SSH hardening은 credential과 관리 경로를 제한하며, TCP Wrapper는 일부 legacy daemon의 host 기반 접근 제어 interface다.

Fail2ban의 동작

authentication log를 정규식 filter로 읽고 일정 시간 안에 실패가 반복된 source IP를 jail action으로 firewall block한다. SSH brute force noise를 줄이지만 distributed attack, spoofable protocol, stolen valid credential, IPv6/forwarded client address 처리를 별도 고려한다.

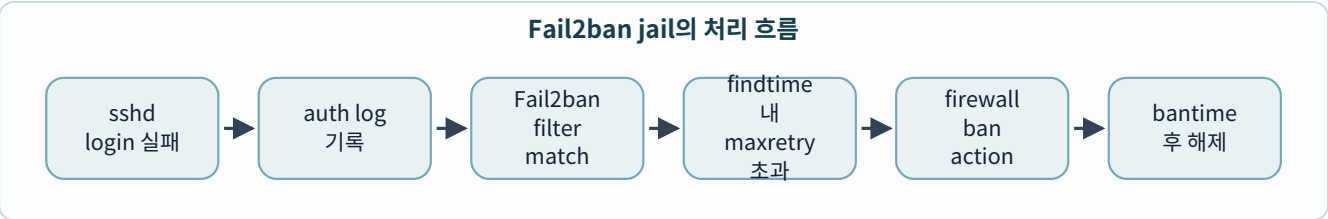


그림 25-2. 로그 path와 timestamp가 틀리거나 reverse proxy source IP를 잃으면 엉뚱한 주소를 차단할 수 있다.

SSH hardening

- password 대신 관리된 public key와 MFA/SSO를 사용한다.
- root direct login을 제한하고 sudo로 개인 identity를 남긴다.
- management network/VPN/bastion과 firewall allowlist로 exposure를 줄인다.
- host key 변경을 monitor하고 client known_hosts 검증을 유지한다.
- 로그인 rate와 실패 event를 중앙 관제한다.

TCP Wrapper

과거 libwrap를 지원하는 daemon에서 /etc/hosts.allow와 hosts.deny로 client address를 제한하던 mechanism이다. 모든 현대 service가 지원하지 않으며 많은 배포판에서 의존도가 줄었다. firewall, application ACL, systemd socket policy를 확인한다.

Zero Trust·Microsegmentation

필기 확인 + 보충 해설

"내부망이면 안전"이라는 경계를 identity와 policy로 재구성한다

함께 찾을 말: Zero Trust, Microsegmentation, Continuous Verification

Zero Trust·Microsegmentation란 Zero Trust는 network 위치에 암묵적 신뢰를 주지 않고 access마다 identity·device·context를 검증해 최소 권한을 적용하는 architecture이며, microsegmentation은 workload 사이 이동 경로를 세밀하게 분리한다.

Zero Trust의 핵심

network 내부라는 위치만으로 신뢰하지 않고, user·device·workload identity와 context, resource sensitivity를 매 access마다 검증하고 최소 권한을 적용한다. "방화벽을 없앤다"거나 특정 제품을 사면 완성되는 architecture가 아니다.

Microsegmentation

workload·application 단위로 east-west communication을 세밀하게 제한해 한 host 침해가 옆 system으로 확산되는 것을 줄인다. VLAN만이 아니라 host firewall, identity-aware proxy, cloud security group, service mesh policy로 구현할 수 있다.

Policy enforcement와 visibility

identity provider, device posture, policy decision point, enforcement point, continuous telemetry가 함께 필요하다. 업무 흐름을 모르고 default deny를 적용하면 운영 중단이 생기므로 discovery와 staged enforcement가 중요하다.

26장. Netfilter · iptables · nftables · firewalld · UFW: Linux 방화벽의 층

> iptables, nftables, firewalld, UFW를 서로 다른 방화벽 다섯 개로 외우면 실제 packet이 어디에서 검사되는지 이해할 수 없다. kernel의 Netfilter hook이 data plane이고, 사용자 공간 도구는 rule을 구성·관리하는 서로 다른 interface다.

Netfilter·Hook·Table·Chain·Rule

필기 확인 + 보충 해설

사용자 명령보다 먼저 kernel에서 packet이 걷는 길을 본다

함께 찾을 말: Netfilter, PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING, Table, Chain, Rule, Conntrack

Netfilter·Hook·Table·Chain·Rule란 Netfilter는 Linux kernel packet path의 hook에서 firewall·NAT 기능을 수행하는 framework이고, table·chain·rule은 packet 조건과 action을 순서 있게 구성하는 정책 구조다.

Netfilter hook

Linux network stack의 여러 지점에 packet processing callback을 연결하는 kernel framework다. 대표 hook은 PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING이다. routing decision 전후와 local process 송수신, forwarding 경로에 policy·NAT·mangle 기능을 배치한다.

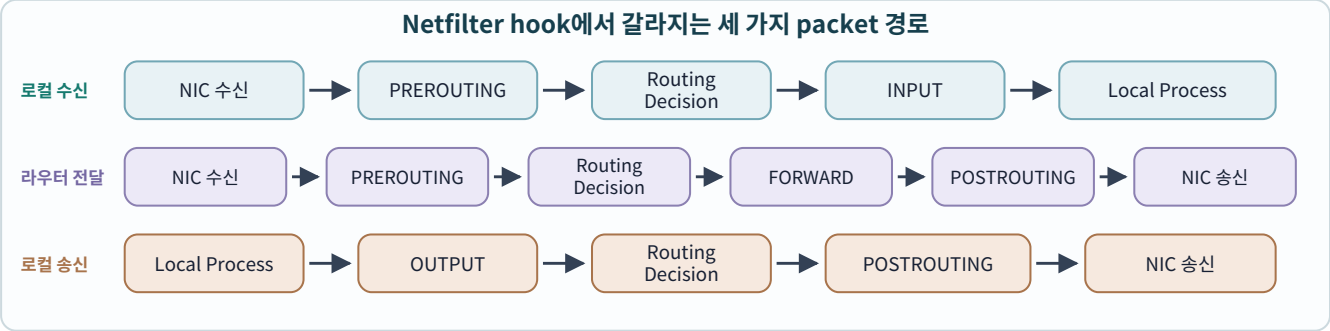


그림 26-1. 수신 packet은 routing decision 뒤 INPUT 또는 FORWARD 중 하나로 갈라지고, local process가 만든 packet은 OUTPUT에서 시작한다. 세 경로 모두 무조건 차례대로 통과하는 것이 아니다.

Table과 Chain

iptables 모델에서 table은 목적별 rule namespace이고 chain은 hook 또는 사용자 정의 rule sequence다. filter는 허용/차단, nat는 connection의 주소 변환 첫 packet 중심, mangle은 field/mark 조정, raw는 connection tracking 예외 같은 저수준 처리를, security는 LSM integration을 담당한다. nftables는 family/table/chain을 더 유연하게 구성한다.

Rule과 first match

chain의 rule은 조건과 action/target으로 구성되며 순서대로 평가한다. ACCEPT/DROP처럼 terminal verdict가 나오면 해당 chain 처리가 끝난다. counter로 실제 match를 검증해야 한다.

Connection Tracking

conntrack은 5-tuple과 TCP 상태 등을 추적해 ESTABLISHED/RELATED 같은 stateful policy와 NAT return mapping을 가능하게 한다. table capacity가 고갈되면 정상 connection도 실패할 수 있어 timeout과 sizing을 monitor한다.

iptables · nftables · firewalld · UFW

필기 확인 + 보충 해설

같은 packet path를 관리하는 서로 다른 추상화 수준

함께 찾을 말: iptables, nftables, nft, firewalld, UFW, Zone, Rich Rule

iptables · nftables · firewalld · UFW란 iptables와 nftables는 kernel firewall rule을 구성하는 interface이고, firewalld와 UFW는 zone · service · 간단 문법으로 그 정책을 상위에서 관리하는 도구다.

iptables와 nftables

iptables는 legacy user-space command와 kernel xtables interface를 중심으로 table/chain rule을 관리한다. nftables는 nft command와 nf_tables kernel subsystem으로 IPv4/IPv6/bridge를 통합하고 set, map, atomic transaction을 제공한다. 현대 배포판의 iptables command가 내부적으로 nft backend를 사용할 수도 있다.

firewalld

zone, service, port, rich rule, policy object를 제공하는 dynamic firewall manager다. runtime configuration과 permanent configuration을 구분하고 backend로 nftables를 흔히 사용한다. interface/source를 zone에 연결해 신뢰 수준별 policy를 관리한다.

UFW

Uncomplicated Firewall은 allow/deny 문법과 application profile로 host firewall 관리를 단순화한다. backend는 배포판에 따라 iptables/nftables 계층을 사용한다. 간단한 interface가 underlying policy를 없애는 것은 아니므로 generated rule과 IPv6 적용을 확인한다.

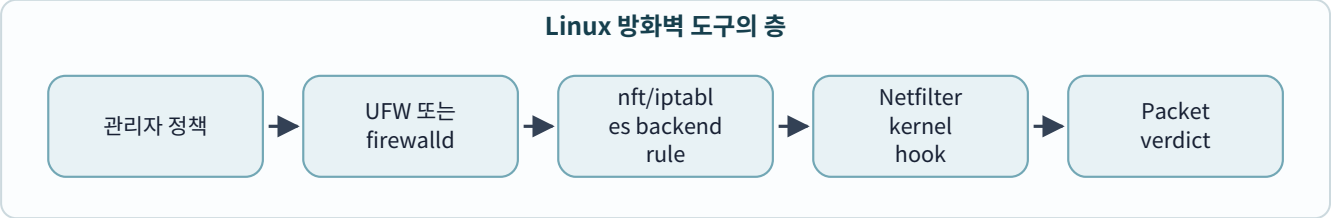


그림 26-2. 여러 관리 도구를 동시에 수정하면 서로 rule을 덮거나 예상하지 못한 순서를 만들 수 있다.

도구	주된 역할	장점
nft	nf_tables 직접 rule language	set/map, 통합 family, atomic update
iptables	legacy rule interface	오래된 문서·환경과 호환
firewalld	zone/service 기반 manager	runtime 변경과 추상화
UFW	간단한 host policy CLI	초기 host firewall 구성 편의

Stateful Firewall · NAT Rule · Policy 검증

필기 확인 + 보충 해설

설정이 존재하는지보다 실제 packet이 어느 rule에 match되는지 확인한다

함께 찾을 말: Stateful Firewall, ESTABLISHED, RELATED, Masquerade, DNAT, SNAT, Firewall Rich Rule

Stateful Firewall · NAT Rule · Policy 검증란 stateful firewall은 connection 상태와 packet header를 함께 판단하고, NAT rule은 특정 hook에서 주소·port를 변환하며, 정책 검증은 listen · route · counter · capture를 순서대로 대조한다.

Stateful allow 예시

외부에서 SSH와 HTTPS 새 연결만 허용하고, 이미 설정된 connection의 reply는 ESTABLISHED로 허용하며, invalid packet은 drop할 수 있다. rule 순서에서 loopback, established, 필요한 service, default drop을 설계한다.

nftables host input policy

```
# nftables 개념 예시
nft add table inet filter
nft 'add chain inet filter input { type filter hook input priority 0; policy drop; }'
nft add rule inet filter input iif lo accept
nft add rule inet filter input ct state established,related accept
nft add rule inet filter input tcp dport { 22, 443 } ct state new accept
nft add rule inet filter input ip protocol icmp accept
nft list ruleset
```

NAT 예시

source NAT은 POSTROUTING에서 나가는 address를 바꾸고, destination NAT은 routing 전 PREROUTING 또는 local OUTPUT에서 destination을 바꾼다. 실제 hook priority와 route 재평가를 고려한다.

상위 관리 도구 예시

```
# firewalld 예시
firewall-cmd --get-active-zones
firewall-cmd --zone=public --add-service=https --permanent
firewall-cmd --zone=public --add-rich-rule='rule family=ipv4 source address=203.0.113.0/24
service name=ssh accept' --permanent
firewall-cmd --reload
firewall-cmd --zone=public --list-all

# UFW 예시
ufw default deny incoming
ufw allow from 203.0.113.0/24 to any port 22 proto tcp
ufw allow 443/tcp
ufw status verbose
```

확인 순서

1. process가 실제 address/port에 listen하는지 ss로 본다.
2. route와 interface가 맞는지 ip route get으로 본다.
3. host firewall rule · counter · log를 본다.
4. upstream security group/router ACL/NAT를 본다.
5. packet capture로 어느 지점까지 SYN이 도착하고 reply가 나가는지 본다.

방화벽을 끄고 해결하는 습관

잠깐의 원인 분리 실험은 격리 lab에서 가능하지만 production에서 전체 firewall disable은 공격면을 열고 원인도 숨긴다. 필요한 flow를 5-tuple과 방향으로 정의해 최소 rule을 추가한다.

27장. NIC에서 Socket까지: Linux가 packet을 받는 실제 경로

> Wireshark가 보이는 packet과 application이 읽는 byte 사이에는 NIC DMA, RX ring, interrupt, NAPI polling, driver, sk_buff, protocol stack, socket buffer가 있다. 이 경로를 알면 packet drop, checksum offload, capture 위치, IDS performance를 설명할 수 있다.

필기 확인 + 보충 해설

NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

packet 한 개가 CPU와 kernel을 통과하는 단계

함께 찾을 말: RX Queue, RX Ring, Descriptor, RX Buffer, Interrupt, Polling, NAPI, NIC Driver, Kernel Network Stack, skb, sk_buff, Socket Receive Buffer

NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer란 NIC는 DMA와 RX descriptor ring으로 frame을 memory에 기록하고, driver와 NAPI가 이를 sk_buff로 network stack에 올리며, transport 처리 후 socket receive buffer에서 application이 읽는다.

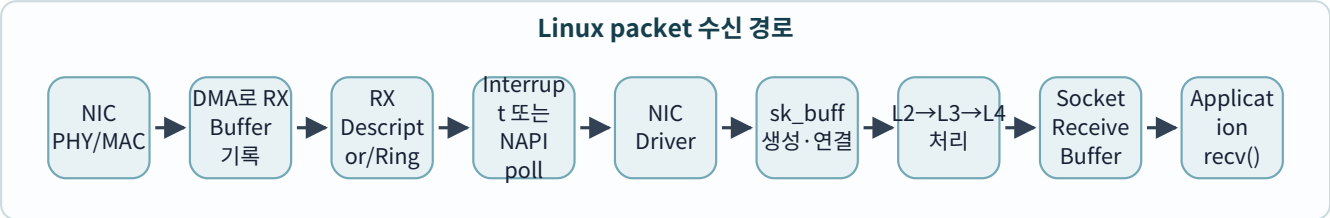


그림 27-1. 고속에서는 packet마다 interrupt하지 않고 NAPI budget만큼 polling해 interrupt storm을 줄인다.

RX ring과 descriptor

driver가 NIC에 사용할 memory buffer 주소를 descriptor ring에 제공한다. NIC는 frame을 DMA로 buffer에 쓰고 descriptor 상태를 완료로 바꾼다. CPU는 producer/consumer index를 따라 완료 항목을 회수한다. ring이 가득 차고 CPU가 제때 처리하지 못하면 NIC-level drop이 발생할 수 있다.

Interrupt와 NAPI

낮은 traffic에서는 interrupt가 빠른 반응을 제공하지만 고속 packet마다 interrupt하면 CPU가 handler 전환에 소모된다. NAPI는 interrupt로 activity를 알린 뒤 interrupt를 완화하고 kernel poll loop가 budget만큼 packet을 처리해 throughput을 높인다. queue가 비면 interrupt를 다시 활성화한다.

sk_buff

Linux kernel의 sk_buff(skb)는 packet data 자체와 header 위치, device, protocol, checksum, routing, timestamp 등 metadata를 가리키는 핵심 구조체다. 모든 단계에서 packet 전체를 복사하는 것이 아니라 buffer와 pointer/metadata를 조정해 처리할 수 있다.

Socket receive buffer

TCP reassembly와 ordering을 거친 byte가 socket receive queue에 쌓이고 application의 recv/read가 이를 가져간다. application이 느리면 buffer가 차고 TCP advertised window가 줄거나 UDP datagram이 drop될 수 있다.

IRQ · RSS · Scheduler · Hyper-Threading

필기 확인 + 보충 해설

고속 packet 처리에서 한 CPU만 바빠지는 이유

함께 찾을 말: IRQ, RSS, CPU Affinity, Scheduler, Hyper-Threading, SMT, Superscalar

IRQ · RSS · Scheduler · Hyper-Threading란 IRQ는 device event를 CPU에 알리고, RSS와 multi-queue는 flow를 여러 queue/CPU에 분산하며, scheduler와 SMT는 process · softirq가 CPU 실행 자원을 공유하는 방식을 결정한다.

IRQ와 CPU affinity

hardware interrupt는 CPU 실행 흐름을 바꿔 device event를 처리하게 한다. multi-queue NIC는 queue별 IRQ를 여러 CPU에 분산할 수 있고 RSS hash로 flow를 queue에 배치한다. 잘못된 affinity는 한 core 포화나 cache locality 저하를 만든다.

Hyper-threading과 scheduler

SMT/Hyper-Threading은 한 physical core의 execution resource를 여러 logical CPU가 공유해 stall 시간을 활용한다. 두 logical CPU가 완전한 두 core 성능을 제공하는 것은 아니다. scheduler는 runnable task를 CPU에 배치하며 softirq와 user process 경쟁을 관찰해야 한다.

Superscalar

CPU가 한 cycle에 여러 instruction을 서로 다른 execution unit에서 처리할 수 있는 구조다. packet processing 성능은 instruction 수뿐 아니라 branch miss, cache miss, memory bandwidth, lock contention의 영향을 받는다.

수신 병목 관찰

```
cat /proc/interrupts
ethtool -l eth0
ethtool -S eth0 | egrep -i 'drop|miss|error|rx'
cat /proc/net/softnet_stat
ss -m
nstat
```

“CPU가 호출된다”의 정확한 표현

NIC event가 interrupt를 발생시키거나 NAPI scheduling을 유도하고, kernel이 해당 CPU에서 driver/softirq code를 실행한다. NIC가 일반 함수처럼 CPU를 직접 호출하는 것은 아니다.

Offloading · Checksum 미완성·Promiscuous Mode

필기 확인 + 보충 해설

host capture가 실제 wire packet과 달라 보이는 이유

함께 찾을 말: Checksum Offload, TSO, GSO, GRO, LRO, Promiscuous Mode, Checksum Incorrect

Offloading · Checksum 미완성·Promiscuous Mode란 network offloading은 segmentation · checksum · packet coalescing을 NIC/kernel에 맡겨 CPU 부담을 줄이고, promiscuous mode는 NIC의 destination MAC 수신 filter를 완화한다.

Checksum offloading

TX checksum offload에서는 OS가 checksum 계산을 NIC에 맡겨 capture 시점의 packet header checksum field가 아직 완성되지 않아 Wireshark가 incorrect라고 표시할 수 있다. 실제 wire에서는 NIC가 계산해 정상일 수 있다. RX offload는 NIC가 검증 결과 metadata를 전달한다.

TSO/GSO와 GRO/LRO

송신 측 TSO/GSO는 큰 buffer를 kernel/NIC가 MTU 크기 segment로 나누게 하고, 수신 측 GRO/LRO는 여러 packet을 큰 단위로 합쳐 stack overhead를 줄인다. host capture에서는 wire와 다른 큰 “packet”처럼 보일 수 있다.

Promiscuous mode

NIC가 자기 MAC, broadcast, 가입 multicast 외의 frame도 OS에 전달하도록 filter를 완화한다. switched network에서 다른 port의 unicast가 자동으로 내 port에 오는 것은 아니므로 promiscuous mode만으로 LAN 전체를 sniff할 수 없다. SPAN/TAP, hub, ARP spoofing 같은 별도 traffic visibility가 필요하다.

Offload와 promiscuous mode 확인

```
ethtool -k eth0
sudo ethtool -K eth0 tx off rx off gro off gso off tso off  # 실습 확인용, 성능 영향
ip link show eth0
ip link set eth0 promisc on
tcpdump -ni eth0 -vvv
```

Offload를 끄는 시점

capture 해석 검증을 위해 lab에서 잠깐 끌 수 있지만 production high-throughput host에서는 CPU 부하와 성능이 크게 변할 수 있다. capture point와 offload 상태를 보고 해석하는 것이 먼저다.

BPF·eBPF·AF_PACKET·DPDK

필기 확인 + 보충 해설

packet을 application까지 모두 복사하기 전에 선별·처리하는 기술

함께 찾을 말: Berkeley Packet Filter, BPF, eBPF, AF_PACKET, DPDK, BPFDoor

BPF·eBPF·AF_PACKET·DPDK란 BPF/eBPF는 kernel hook에서 검증된 program으로 packet·system event를 filter·관찰하고, AF_PACKET은 L2 packet socket, DPDK는 kernel을 우회한 user-space 고속 packet processing framework다.

Classic BPF와 eBPF

BPF는 원래 packet capture filter bytecode로 시작해 kernel에서 불필요 packet을 일찍 거르는 mechanism이었다. eBPF는 verifier를 통과한 sandboxed program을 network, tracing, security hook에 attach해 map과 helper로 state를 다룰 수 있게 확장했다.

AF_PACKET

Linux packet socket family로 L2 frame을 user space에 전달한다. tcpdump/libpcap, IDS capture backend가 사용할 수 있고 TPACKET ring으로 memory-mapped high-speed capture를 제공한다.

DPDK

Data Plane Development Kit은 user-space application이 NIC queue를 polling하고 hugepage·zero-copy를 활용해 kernel network stack을 우회하는 고속 packet processing framework다. 전용 core와 NIC binding, 보안·운영 복잡성이 따른다. 필기의 “dfdk”는 DPDK를 가리킬 가능성이 높다.

BPFdoor의 원리 수준 이해

BPFdoor로 알려진 Linux backdoor는 raw packet socket과 BPF filter를 사용해 일반 listen port 없이 특정 magic packet을 감시하고, 조건이 맞으면 remote control channel을 여는 방식으로 분석되었다. BPF 자체가 malware인 것이 아니라 합법 packet filter 기능을 은폐·trigger에 악용한 사례다.

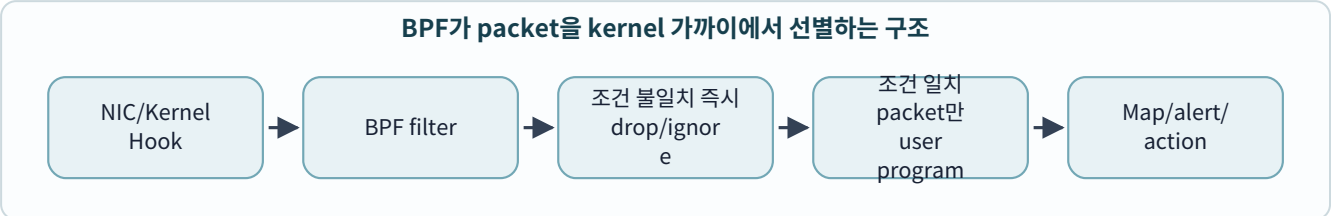


그림 27-2. capture 성능을 높이는 정상 기능과 covert trigger를 감시하는 공격 사용은 목적과 program provenance로 구분한다.

28장. SPAN·TAP·Inline·DAQ: 보안 센서가 traffic을 보는 방법

> IDS rule이 정확해도 sensor가 packet을 못 받거나 packet loss가 생기면 탐지할 수 없다. 보안 관제의 시작은 traffic visibility다. port mirroring, TAP, inline bridge가 sensor 장애와 packet 경로에 미치는 영향을 비교한다.

SPAN · Port Mirroring · TAP

필기 확인 + 보충 해설

센서가 무엇을 보며 어떤 packet을 놓칠 수 있는가

함께 찾을 말: SPAN, Switched Port Analyzer, Port Mirroring, Local SPAN, RSPAN, ERSPAN, Network TAP

SPAN · Port Mirroring · TAP란 SPAN은 switch가 선택한 traffic 복사본을 monitoring port로 보내는 기능이고, TAP은 물리 link에서 traffic 복사본을 제공하는 장치다.

SPAN

Switched Port Analyzer는 switch의 source port/VLAN traffic 복사본을 destination monitoring port로 보내는 port mirroring 기능이다. local SPAN, remote SPAN(RSPAN), encapsulated remote SPAN(ERSPAN) 같은 형태가 있다.

SPAN의 한계

source 총 traffic이 destination port capacity를 넘으면 복사본 drop이 생길 수 있다. switch가 bad frame을 이미 폐기하거나 timestamp가 정확하지 않을 수 있고, oversubscription과 configuration 실수로 visibility gap이 생긴다. sensor는 원본 forwarding에 직접 영향이 적다는 장점이 있다.

Network TAP

물리 link 사이에 배치해 양방향 signal/packet의 복사본을 monitoring port로 제공하는 장치다. passive optical TAP, active copper TAP, aggregation TAP 등이 있다. 장애 모드, power loss, link negotiation, 양방향 traffic 합산 용량을 확인한다.

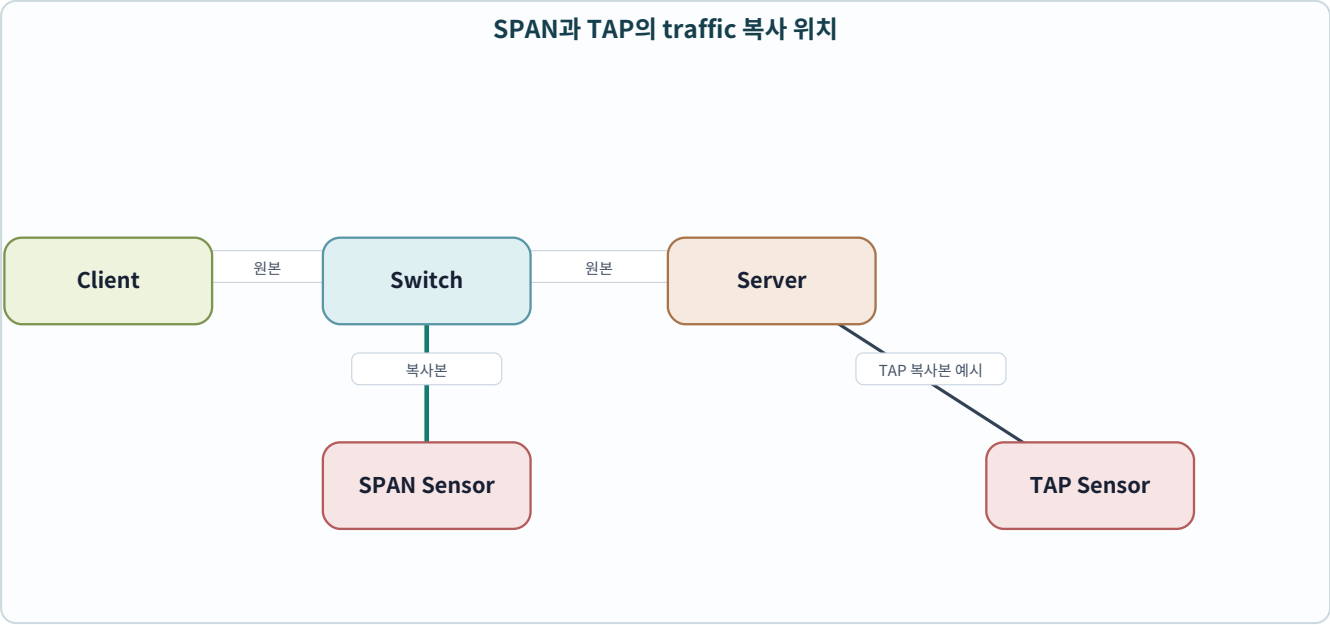


그림 28-1. sensor는 복사본을 받는 out-of-band 구조라 차단하려면 별도 연동이 필요하다.

Inline/Non-inline·Bridge/Router Mode·UTM/NGFW

필기 확인 + 보충 해설

탐지 장비가 service availability에 미치는 영향

함께 찾을 말: Inline IDS/IPS, Non-inline, Bridge Mode, Router Mode, UTM, NGFW, Fail-open, Fail-close

Inline/Non-inline·Bridge/Router Mode·UTM/NGFW란 inline sensor는 원본 traffic path에서 차단할 수 있고 non-inline sensor는 복사본을 관찰하며, bridge mode는 L2 투명 전달, router mode는 L3 routing을 수행한다.

Inline과 Non-inline

inline sensor/IPS는 traffic path 가운데 있어 packet이 sensor를 통과해야 하므로 즉시 drop·modify할 수 있지만 latency와 장애 위험이 있다. non-inline IDS는 복사본을 분석해 원본 traffic을 직접 막지 못하지만 sensor failure가 service path를 끊지 않는다.

Bridge mode와 Router mode

transparent bridge inline 장비는 IP topology를 크게 바꾸지 않고 L2로 두 port를 연결하면서 검사한다. router mode는 L3 interface와 routing을 수행한다. UTM/NGFW는 firewall, IPS, VPN, application control 등 여러 기능을 통합하며 mode에 따라 deployment가 달라진다.

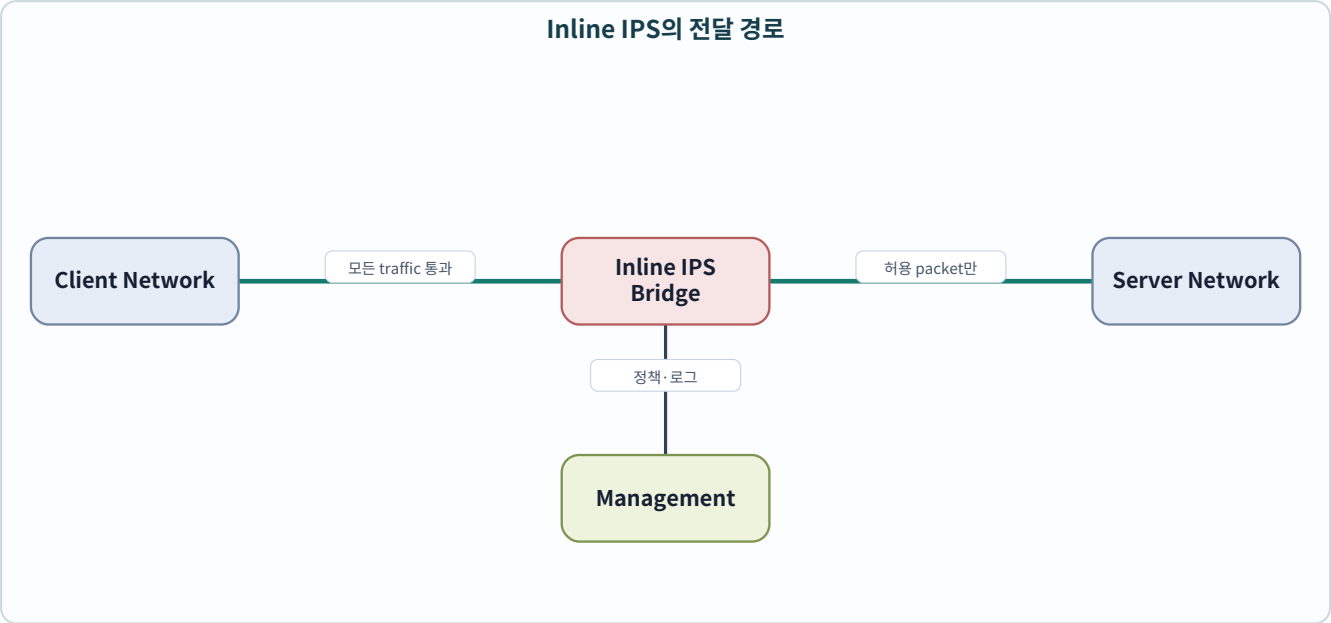


그림 28-2. fail-open은 장애 시 통신 지속, fail-close는 차단을 우선한다. service와 risk에 따라 bypass 설계를 결정한다.

UTM과 NGFW

UTM은 여러 security 기능을 하나의 appliance로 통합한다는 운영 관점, NGFW는 stateful firewall에 application/user awareness, IPS, TLS inspection 등을 확장한 시장 용어다. 제품 기능은 label보다 실제 inspection path와 license를 확인한다.

DAQ · PCAP · Packet Capture

필기 확인 + 보충 해설

rule engine 앞의 packet 수집 계층

함께 찾을 말: DAQ, Data Acquisition, PCAP, pcapng, libpcap, tcpdump

DAQ · PCAP · Packet Capture란 DAQ는 IDS engine과 packet input backend를 연결하는 acquisition layer이고, PCAP은 captured packet과 timestamp · link metadata를 저장·처리하는 표준 생태계다.

DAQ

Snort의 Data Acquisition library 계층은 packet source와 engine 사이를 추상화해 pcap, AFPacket, NFQ, inline backend 등에서 packet을 받고 verdict를 적용한다. 정확한 module 지원은 Snort version과 build를 확인한다.

PCAP와 libpcap

PCAP은 packet capture file format과 capture ecosystem을 가리키며 libpcap은 OS capture API를 공통 interface로 제공한다. timestamp, snaplen, link type, packet length가 file에 저장된다. pcapng는 interface와 metadata를 더 풍부하게 표현한다.

Packet loss 측정

NIC drop, kernel ring drop, capture library drop, engine packet drop를 각각 counter로 확인해야 한다. sensor가 1Gbps port에 여러 1Gbps source를 mirror받으면 wire speed 합계가 초과될 수 있다.

Capture loss 확인 포인트

```
# capture 기본
sudo tcpdump -ni eth0 -s 0 -w capture.pcap
capinfos capture.pcap

# interface 통계
ip -s link show eth0
ethtool -S eth0

# Suricata stats / Snort perf output은 제품 설정에 맞춰 확인
```

29장. IDS·IPS·Snort·Suricata·Zeek·Security Onion

> IDS는 packet을 보고 “공격 이름을 맞히는 기계”가 아니다. signature, protocol state, file metadata, behavior baseline, context를 이용해 관찰 event를 만들고 analyst가 사건으로 판단할 evidence를 제공한다.

IDS·IPS·HIDS·NIDS·탐지 방식

필기 확인 + 보충 해설

탐지 위치와 차단 권한, 판단 근거를 분리한다

함께 찾을 말: IDS, IPS, HIDS, NIDS, Signature-based Detection, Heuristic-based Detection, False Positive, False Negative

IDS·IPS·HIDS·NIDS·탐지 방식란 IDS는 system/network event를 분석해 suspicious activity를 alert하고, IPS는 traffic path에서 차단까지 수행하며, HIDS/NIDS는 관찰 위치가 host/network로 다르고 signature와 heuristic은 탐지 근거가 다르다.

HIDS와 NIDS

HIDS는 endpoint의 process, file, registry, system call, log를 관찰하고, NIDS는 network packet과 flow를 관찰한다. network에서 encrypted payload는 제한되고, host에서는 process context를 보지만 설치·우회 위험이 있다. 둘을 결합한다.

IDS와 IPS

IDS는 주로 detect·alert하고, IPS는 inline 또는 enforcement 연동으로 packet/session을 차단할 수 있다. IPS rule은 false positive가 service outage로 이어질 수 있어 confidence, exception, staged deployment가 필요하다.

Signature와 Heuristic

signature-based detection은 알려진 byte pattern, protocol field, exploit behavior를 rule로 match해 설명 가능하고 정확할 수 있지만 변형·미지 공격에 약하다. heuristic/anomaly는 정상 baseline이나 suspicious behavior를 찾아 unknown을 볼 수 있지만 tuning과 false positive가 많다.



그림 29-1. 한 alert의 존재보다 원본 packet, flow, host process, user context를 함께 본다.

False Positive와 False Negative

false positive는 정상인데 공격으로 탐지, false negative는 공격인데 놓침이다. threshold를 높이면 alert 양은 줄지만 놓침이 늘 수 있다. precision, recall, base rate와 운영 처리 능력을 함께 고려한다.

Snort/Suricata Rule·SID·rev·flow·PCRE

필기 확인 + 보충 해설

문자열 하나를 찾는 것보다 stream과 protocol context를 검사한다

함께 찾을 말: Snort Rule, Suricata Rule, Ruleset, SID, rev, flow, TCP Follow, PCRE, Fragmentation Attack, Teardrop

Snort/Suricata Rule·SID·rev·flow·PCRE란 IDS rule은 traffic header 조건과 protocol-aware option을 조합해 event를 정의하고, SID는 rule identity, rev는 version, flow는 connection 방향·상태, PCRE는 정규식 pattern을 지정한다.

Rule header와 option

Snort/Suricata 계열 rule은 action, protocol, source/destination address·port, direction의 header와 msg, flow, content, pcre, sid, rev 등의 option으로 구성된다. engine version과 keyword 지원이 다를 수 있다.

교육용 rule 구조 예시

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (
  msg:"LAB suspicious admin path";
  flow:to_server,established;
  http.uri; content:"/admin/debug"; nocase;
  sid:1000001; rev:2;
)
```

SID와 rev

SID는 rule의 고유 identifier이고 rev는 같은 SID rule 내용의 revision이다. local SID range는 조직 convention과 engine 문서를 따른다. 동일 SID 충돌은 rule 관리와 alert 해석을 망친다.

flow와 TCP follow

flow keyword는 connection direction과 established state를 제한한다. TCP stream reassembly는 segment가 나뉘거나 순서가 바뀐 payload를 logical stream으로 복원해 content를 검사한다. 단순 packet 하나만 보면 evasion을 놓칠 수 있다.

PCRE

Perl Compatible Regular Expressions는 복잡한 pattern을 표현하지만 CPU 비용이 크고 catastrophic backtracking 위험이 있다. 가능하면 빠른 content anchor로 후보를 줄인 뒤 regex를 적용한다.

Fragmentation attack와 normalization

IP fragment를 겹치거나 비정상 offset으로 보내 endpoint와 IDS의 reassembly 해석 차이를 악용할 수 있다. engine은 target OS policy와 overlap 처리, timeout을 맞추고 fragment를 재조립한 뒤 상위 protocol을 검사한다. Teardrop은 overlapping fragment 처리 취약점을 악용한 역사적 공격이다.

Snort·Suricata·Zeek

필기 확인 + 보충 해설

signature alert와 풍부한 protocol telemetry를 함께 쓴다

함께 찾을 말: Snort, Suricata, Zeek, Bro, Lua

Snort·Suricata·Zeek란 Snort와 Suricata는 rule과 protocol inspection을 이용한 IDS/IPS engine이고, Zeek는 network transaction을 고수준 event와 structured log로 변환하는 analysis framework다.

Snort

signature 중심 NIDS/IPS engine으로 rule language와 preprocessors/inspectors를 통해 packet·stream을 분석한다. Snort 2와 Snort 3의 architecture와 configuration이 다르다.

Suricata

multi-threaded IDS/IPS/NSM engine으로 protocol parser, file extraction, EVE JSON event, AF_PACKET/NFQ/DPDK 등 다양한 run mode를 제공한다. rule 호환성이 높아도 keyword와 behavior 차이를 test한다.

Zeek

Zeek는 packet마다 signature alert만 내기보다 connection, DNS, HTTP, TLS, file 같은 고수준 transaction log와 event scripting을 생성하는 network analysis framework다. “정상인데 이상한 행동”을 hunting하기에 풍부한 metadata를 제공한다.

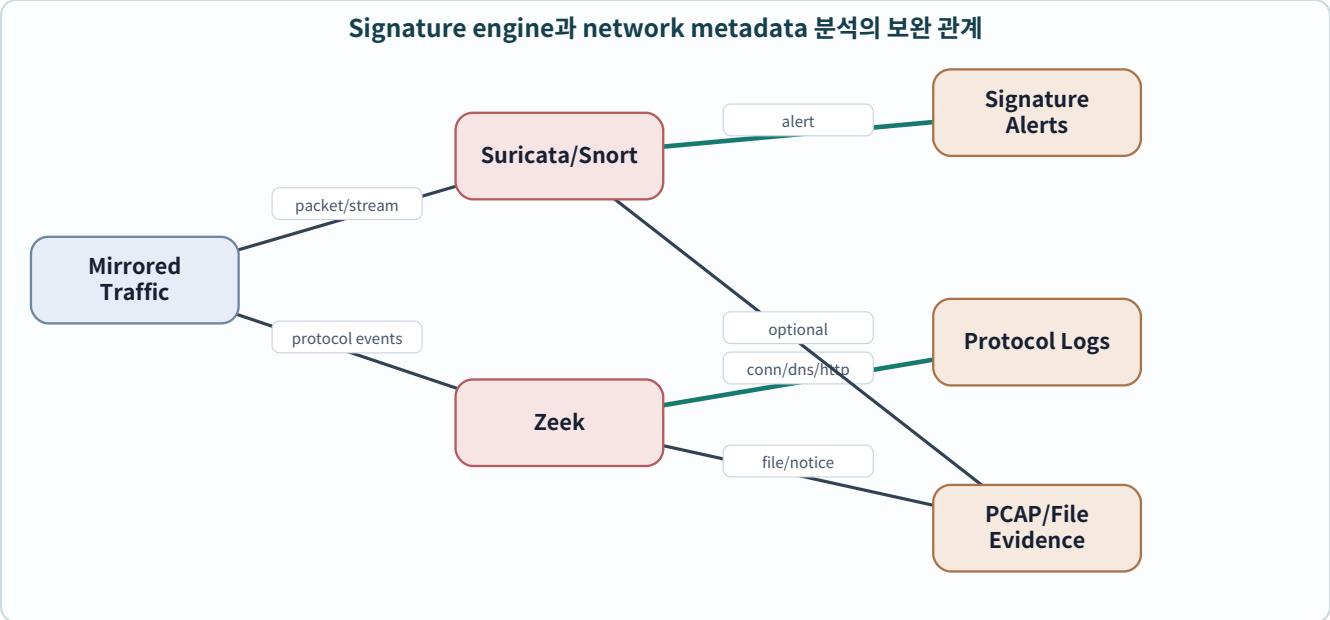


그림 29-2. 같은 sensor traffic에서 서로 다른 evidence를 만들어 SIEM에서 연결한다.

Lua

Lua는 가벼운 embedded scripting language로 IDS/packet tool에서 custom detection·extension에 쓰일 수 있다. engine별 API와 sandbox 제한을 확인한다.

NSM·Security Onion·Sguil/Squert/Barnyard2

필기 확인 + 보충 해설

도구 묶음보다 evidence 수집과 analyst workflow가 중심

함께 찾을 말: Network Security Monitoring, NSM, Security Onion, Barnyard2, Sguil, Squert

NSM·Security Onion·Sguil/Squert/Barnyard2란 NSM은 network telemetry와 packet evidence로 threat를 지속 관찰·조사하는 운영 방식이고, Security Onion은 이를 위한 sensor·analysis component를 통합한 platform이다.

NSM

Network Security Monitoring은 network traffic에서 session, alert, transaction log, packet evidence를 수집해 threat를 탐지·조사하는 운영 discipline이다. tool 설치보다 sensor placement, data quality, analyst workflow가 핵심이다.

Security Onion

NSM distribution/platform으로 Suricata, Zeek, Elastic 계열, packet capture, case workflow 등을 통합한다. version에 따라 component와 architecture가 달라져 legacy Sguil, Squert, Barnyard2 설명을 현재 구성과 구분해야 한다.

Barnyard2·Sguil·Squert

Barnyard2는 Snort unified2 output을 읽어 DB/consumer로 전달하던 spooler, Sguil은 analyst console, Squert는 web alert visualization 도구로 legacy Security Onion 교육에서 자주 등장한다. 현재 제품 version에서는 다른 pipeline으로 대체되었을 수 있다.

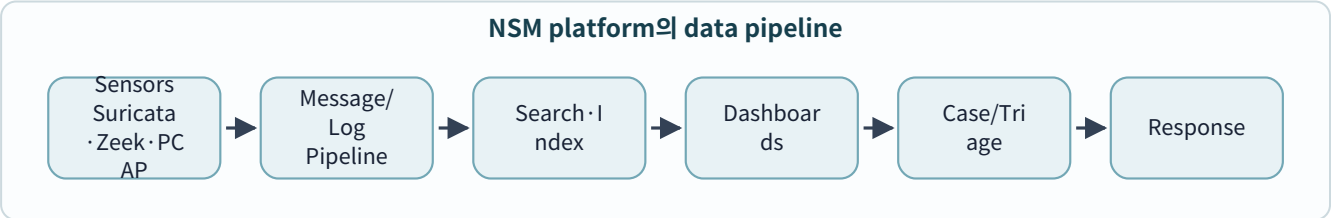


그림 29-3. 제품 이름이 바뀌어도 sensor→normalize→store→investigate의 구조는 유지된다.

Security Onion에서 확인할 것

- sensor interface에 IP를 주지 않는 sniffing design과 management interface 분리
- packet drop과 disk retention, index lifecycle
- time synchronization과 hostname/site metadata

- rule update, local rule version control, exception 기록
- case evidence와 원본 PCAP 보존 정책

30장. SIEM · EDR · XDR · SOAR · MITRE ATT&CK: Alert를 대응으로 연결하기

> 관제는 log를 한 화면에 모으는 것이 아니라, 여러 sensor의 event를 공통 시간과 entity로 연결해 위험한 행동 sequence를 찾고, 조사·containment·복구·개선까지 닫힌 loop로 운영하는 일이다.

SIEM · ELK · Wazuh

필기 확인 + 보충 해설

로그 저장소를 탐지 use case와 사건 조사 도구로 만든다

함께 찾을 말: SIEM, ELK Stack, Elasticsearch, Logstash, Kibana, Wazuh, Correlation

SIEM · ELK · Wazuh란 SIEM은 여러 system의 event를 수집·normalize · correlate해 탐지와 조사를 지원하고, ELK는 검색·수집·시각화 stack, Wazuh는 endpoint telemetry와 security rule을 결합한 platform이다.

SIEM

Security Information and Event Management는 여러 source의 log/event를 수집·normalize · index하고 correlation, search, dashboard, alert, retention을 제공한다. “모든 log를 넣는다”보다 use case에 필요한 field와 data quality, cost를 설계한다.

ELK Stack과 Wazuh

Elasticsearch는 distributed search/index, Logstash는 ingest/transform pipeline, Kibana는 visualization을 제공하는 ELK stack의 중심이다. Wazuh는 endpoint agent, file integrity, vulnerability/configuration monitoring, rule/decoder와 Elastic/OpenSearch 계열 backend를 조합하는 security platform이다. version에 따라 architecture가 달라진다.

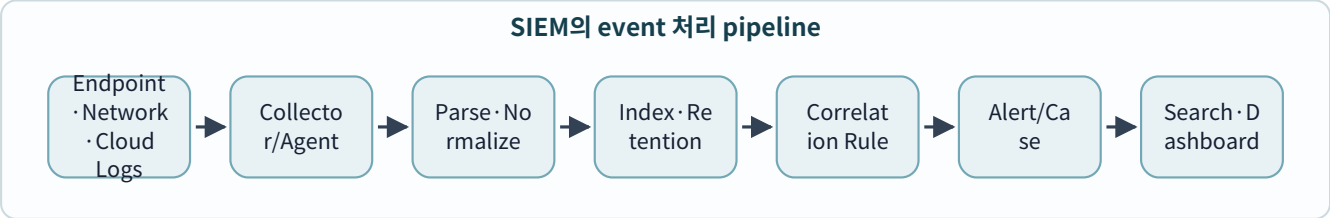


그림 30-1. 원본 event와 normalized field를 함께 보존해야 parser 오류와 investigation을 검증할 수 있다.

Use case와 rule lifecycle

예: “새벽에 privileged account가 여러 host에 로그인 후 security tool을 중지”를 필요한 log source, field, threshold, exception, response owner로 정의한다. rule은 test data, false positive review, version, coverage metric으로 관리한다.

EDR·XDR·SOAR·능동 대응

필기 확인 + 보충 해설

탐지 화면에서 실제 환경 변경까지의 권한 chain

함께 찾을 말: EDR, XDR, SOAR, Active Response, Playbook, Host Isolation

EDR·XDR·SOAR·능동 대응란 EDR은 endpoint behavior를 수집·조사·격리하고, XDR은 여러 security domain의 telemetry를 연결하며, SOAR는 enrichment와 response action을 playbook으로 orchestration한다.

EDR

Endpoint Detection and Response는 endpoint agent/sensor로 process creation, command line, file, registry, network connection, memory behavior를 수집하고 hunting, isolation, process kill, file quarantine 같은 response를 제공한다. antivirus signature보다 behavior와 investigation context를 확장한다.

XDR

Extended Detection and Response는 endpoint뿐 아니라 identity, email, network, cloud telemetry를 하나의 detection/response model로 연결한다는 제품·architecture 범주다. vendor별 범위가 크게 다르므로 실제 sensor, open API, retention, response authority를 확인한다.

SOAR

Security Orchestration, Automation and Response는 alert enrichment, ticket, approval, containment action을 playbook으로 연결한다. 자동화가 잘못된 data를 믿으면 정상 account를 대량 차단할 수 있어 confidence, human approval, rollback, audit를 설계한다.



그림 30-2. 능동 대응은 빠르지만 action 권한과 오탐 영향이 크므로 단계적 승인과 복구 절차가 필요하다.

Active Response

탐지 후 source block, account disable, host isolation, DNS sinkhole 등 환경을 변경하는 대응이다. 공격자에게 탐지 사실을 알리고 evidence를 잃거나 business outage를 만들 수 있어 scope와 authority를 사전에 정한다.

Kill Chain · MITRE ATT&CK · C2

필기 확인 + 보충 해설

alert를 공격 행동 sequence와 방어 coverage로 연결한다

함께 찾을 말: Cyber Kill Chain, MITRE ATT&CK, Tactic, Technique, Procedure, Command and Control, C2, C&C

Kill Chain · MITRE ATT&CK · C2란 Kill Chain은 공격 단계를 선형 관점으로 나눈 모델이고, MITRE ATT&CK은 공격자의 목표와 behavior technique을 지식화하며, C2는 침해된 host와 명령을 교환하는 통신 체계다.

Cyber Kill Chain

공격을 reconnaissance, weaponization, delivery, exploitation, installation, command and control, actions on objectives 같은 단계로 보는 모델이다. 모든 공격이 선형으로 정확히 따르지는 않지만 어느 단계에서 detect·disrupt할지 계획하는 데 도움을 준다.

MITRE ATT&CK

실제 관찰된 adversary behavior를 tactic(목표), technique/sub-technique(방법), procedure(구체 구현)로 정리한 knowledge base다. ATT&CK ID를 alert에 붙이는 것만으로 탐지 coverage가 보장되지 않으며 data source와 analytic validation이 필요하다.



그림 30-3. 공격은 앞뒤로 이동하고 여러 technique을 반복할 수 있다.

C2/C&C Server

Command and Control server는 침해된 agent와 명령·결과·payload를 교환하는 infrastructure다. HTTP(S), DNS, cloud service, custom protocol을 이용해 정상 traffic에 섞일 수 있다. beacon 주기, domain age, JA3/JA4류 fingerprint, uncommon destination, process-network relation을 함께 본다.

Detection engineering

technique를 data source와 measurable behavior로 바꾸고, test procedure로 telemetry가 실제 발생하는지 확인하며, false positive와 evasion을 반복 개선하는 과정이다. atomic test와 purple-team exercise가 validation에 쓰인다.

Incident Response · Evidence · Containment

필기 확인 + 보충 해설

빠른 차단과 정확한 원인 규명을 함께 달성한다

함께 찾을 말: Incident Response, Containment, Eradication, Recovery, Chain of Custody, Evidence Integrity

Incident Response · Evidence · Containment란 incident response는 준비, 탐지·분석, containment, eradication, recovery, 개선의 lifecycle이고, evidence integrity는 조사 자료의 원본성과 시간·처리 이력을 보존하는 원칙이다.

Incident response lifecycle

1. Preparation: role, contact, tool, backup, legal authority 준비.
2. Detection/Analysis: alert를 validate하고 scope · timeline · impact를 판단.
3. Containment: 확산을 제한하되 evidence와 service를 고려.
4. Eradication: persistence, vulnerable account, malware, root cause 제거.
5. Recovery: clean state 검증, service 복구, 강화 monitoring.
6. Lessons learned: control gap, rule, process, documentation 개선.

Evidence integrity

원본 log/PCAP/disk image를 read-only 또는 hash와 chain of custody로 보존하고, 조사 copy에서 분석한다. timestamp source와 timezone을 기록한다. live response는 system 상태를 바꾸므로 실행 command와 순서를 남긴다.

능동 대응의 안전장치

- 차단 대상 entity의 confidence와 asset criticality 확인
- 승인자와 emergency exception 정의
- action timeout과 자동 rollback
- 대응 전후 evidence snapshot
- out-of-band 복구 경로와 break-glass account

31장. Governance·Compliance·CVE·CWE·OWASP·KISA 기준 읽기

> 취약점 이름과 점검 기준을 그대로 동일시하면 “왜 위험한가”와 “어떤 통제가 필요한가”를 놓친다. CVE는 개별 공개 취약점, CWE는 약점 유형, OWASP Top 10은 위험 범주, KISA 가이드는 국내 평가·조치 기준이라는 서로 다른 층이다.

Governance·Compliance

필기 확인 + 보충 해설

체크리스트 충족과 실제 위험 관리의 차이

함께 찾을 말: Governance, Compliance, Policy, Standard, Procedure, Guideline, Risk Appetite

Governance·Compliance란 governance는 조직의 보안 의사결정·책임·risk 방향을 정하고, compliance는 정해진 법·표준·policy 요구를 충족하고 증명하는 활동이다.

Governance

조직의 보안 목표, risk appetite, 책임, 의사결정, 성과 monitoring을 정하는 상위 체계다. board/경영진과 업무 owner가 “무엇을 왜 보호할지” 결정하고 policy와 resource를 배분한다.

Compliance

법률, regulation, contract, standard, 내부 policy의 요구사항을 충족하고 evidence로 증명하는 활동이다. compliance를 통과했다고 모든 실제 threat에 안전한 것은 아니며, governance가 compliance 범위를 risk와 연결해야 한다.

Policy·Standard·Procedure·Guideline

문서	역할
Policy	조직이 지켜야 할 원칙과 책임
Standard	반드시 만족할 구체 기준
Procedure	작업을 수행하는 단계
Guideline	권고되는 좋은 방법

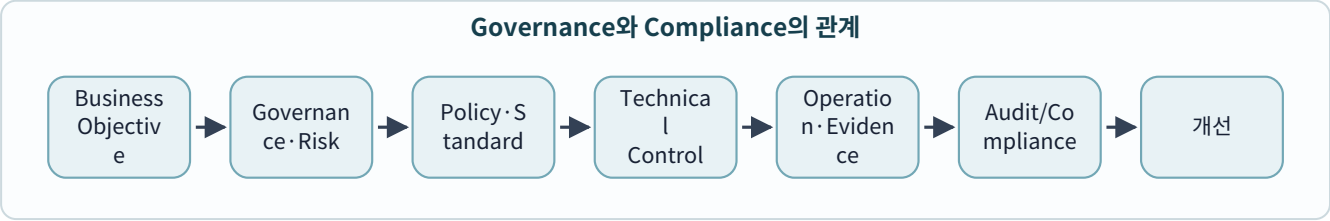


그림 31-1. 감사 evidence는 통제가 존재하는지뿐 아니라 실제로 지속 운영되는지를 보여야 한다.

CVE·CWE·CVSS

필기 확인 + 보충 해설

사건 이름, 원인 유형, 심각도, 조직 위험을 분리한다

함께 찾을 말: CVE, CWE, CVSS, Vulnerability Identifier, Weakness

CVE·CWE·CVSS란 CVE는 개별 공개 취약점의 식별자, CWE는 반복되는 약점 유형의 분류, CVSS는 기술적 severity를 표현하는 점수 체계다.

CVE

공개된 특정 product/version의 vulnerability에 공통 identifier를 부여한다. CVE record는 이름표와 핵심 설명이며 severity, exploitability, patch 정보는 NVD, vendor advisory 등 추가 source를 함께 본다.

CWE

software/hardware 설계·구현에서 반복되는 weakness 유형을 분류한다. 예: improper input validation, out-of-bounds write. 하나의 CVE가 여러 CWE와 관련되거나 정확한 mapping이 불명확할 수 있다.

CVSS

vulnerability의 기술적 severity를 vector로 평가하는 scoring system이다. 조직 자산의 exposure와 business impact를 자동으로 포함하는 risk score는 아니므로 asset context와 active exploitation을 함께 본다.

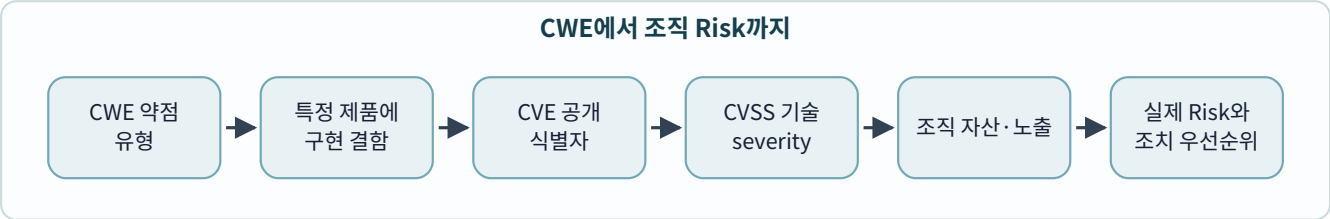


그림 31-2. CVE 번호가 없다고 위험이 없거나, CVSS 10이면 모든 조직에서 무조건 첫 조치라는 뜻은 아니다.

OWASP·KISA·주요정보통신기반시설 가이드

필기 확인 + 보충 해설

risk 범주와 상세 점검 기준을 목적에 맞게 사용한다

함께 찾을 말: OWASP, OWASP Top 10, ASVS, KISA, 주정통, 주요정보통신기반시설 기술적 취약점 분석·평가방법 상세 가이드

OWASP·KISA·주요정보통신기반시설 가이드란 OWASP는 application security 지식 체계와 risk guide를 제공하고, KISA의 기반시설 가이드는 국내 중요 정보시스템의 기술적 취약점 점검 기준과 평가 방법을 제공한다.

OWASP

Open Worldwide Application Security Project는 web/application security 지식과 도구를 공개하는 community다. OWASP Top 10은 주기적으로 선정되는 web application risk 범주 awareness 문서이며 모든 test case를 완전 열거하는 표준은 아니다. ASVS, Testing Guide, Cheat Sheet를 함께 사용한다.

KISA와 주요정보통신기반시설 가이드

KISA는 국내 Internet·information security 관련 기관으로 guide와 취약점 정보를 제공한다. “주요정보통신기반시설 기술적 취약점 분석·평가 방법 상세가이드”는 system, network, security device, DB, web 등 점검 항목과 판단 기준을 제공한다. version과 적용 대상, 법적 근거를 확인한다.

점검 항목과 취약점의 관계

점검 항목은 password policy, 불필요 service, permission처럼 control 상태를 검사하고, CWE/CVE는 software weakness와 개별 defect를 표현한다. 하나의 점검 항목이 여러 weakness를 완화할 수 있고, 특정 CVE는 checklist에 없어도 patch가 필요하다.

OWASP risk와 실습

SQL injection, XSS, broken access control 같은 범주는 원인·영향·방어를 이해하고 승인된 DVWA/WebGoat/bWAPP lab에서 재현한다. production 대상의 공격 payload 실행은 scope와 허가가 필요하다.

보안 기준과 실제 통제 검증

필기 확인 + 보충 해설

문서상 준수에서 실제 방어 효과로 넘어가기

함께 찾을 말: Security Baseline, Control Validation, Compensating Control, Exception Management

보안 기준과 실제 통제 검증란 보안 기준은 필요한 통제를 체계적으로 확인하게 하지만, 실제 architecture와 data flow에 맞춰 적용하고 observable test로 통제 효과를 검증해야 한다.

Security architecture review

network diagram, data flow, identity flow, secret storage, third-party dependency를 보고 trust boundary마다 authentication, authorization, encryption, logging, failure mode를 검토한다. checklist는 질문을 놓치지 않게 하지만 system context를 대신하지 않는다.

Control validation

설정 파일 존재만 확인하지 않고 실제 denied request, failover, restore, alert delivery를 test한다. firewall rule counter, EDR isolation, backup restore time, certificate expiration alert처럼 observable outcome을 확인한다.

Exception management

업무상 기준을 즉시 충족할 수 없으면 owner, reason, scope, compensating control, expiry, review를 기록한다. 영구 예외가 되지 않도록 종료 조건과 risk acceptance authority를 명확히 한다.

PART VI

승인된 모의해킹·웹 보안·통합 실습

모의해킹은 도구를 많이 실행하는 활동이 아니라, 합의된 범위에서 공격 가능성을 검증하고 영향과 재현 조건을 증거로 남긴 뒤 안전한 조치와 재시험으로 마무리하는 보안 평가다. 모든 실습은 자신이 소유하거나 명시적으로 허가받은 격리 환경을 전제로 한다.

32장. 모의해킹 방법론과 OSINT·Scanning·취약점 분석

> Nmap을 실행하기 전에 대상, 시간, 허용 기법, 중단 조건, data 처리 방식을 합의해야 한다. 기술적으로 가능한 것과 평가 계약상 허용된 것은 다르다.

모의해킹·Rules of Engagement

필기 확인 + 보충 해설

공격 도구가 아니라 검증 절차와 책임이 중심

함께 찾을 말: Penetration Testing, Pentest, Rules of Engagement, Scope, Reconnaissance, Exploitation, Retest

모의해킹·Rules of Engagement란 모의해킹은 명시적 승인과 범위 안에서 공격자의 관점으로 security control을 검증하고 evidence·impact·조치안을 제공하는 평가이며, Rules of Engagement는 허용 행위와 중단 조건을 정한 운영 규칙이다.

Rules of Engagement

대상 IP/domain, 제외 system, 시험 시간, source IP, 허용된 exploit·password test·DoS 여부, 담당자, 긴급 중단 연락처, evidence 보관과 삭제를 문서화한다. production에서 availability test와 phishing은 별도 승인 없이 수행하지 않는다.

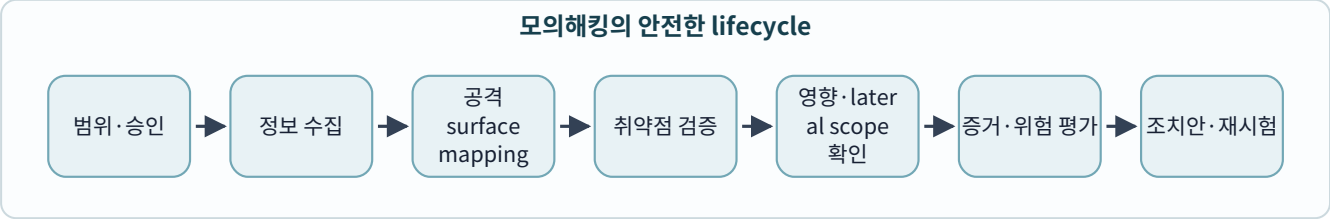


그림 32-1. 각 단계에서 발견한 정보가 다음 단계의 가설이 되며, 필요 이상의 침투는 하지 않는다.

정보 수집과 enumeration

passive OSINT는 대상에 직접 packet을 보내지 않고 공개 source에서 domain, certificate, code repository, 채용 정보 등을 찾는다. active enumeration은 DNS query, port scan, service handshake처럼 대상에 traffic을 발생시킨다. stealth라는 표현보다 탐지 가능성과 업무 영향, 승인 범위를 명확히 한다.

취약점 분석과 exploitation

scanner 결과는 candidate다. version banner 오류, backport patch, configuration 때문에 false positive가 가능하다. exploit은 취약점이 실제 영향을 만드는지 최소한으로 검증하고 shell 획득 자체보다 권한·data exposure·business impact를 기록한다.

보고서

finding마다 제목, 자산, severity 근거, 사전 조건, 재현 단계, evidence, 영향, root cause, 구체 조치, reference, retest 결과를 적는다. secret와 personal data는 최소 수집·마스킹한다.

OSINT·WHOIS·Shodan·Maltego

필기 확인 + 보충 해설

검색 결과를 증거로 만들기 위해 출처와 시점을 검증한다

함께 찾을 말: OSINT, Open Source Intelligence, WHOIS, RDAP, Shodan, Maltego

OSINT·WHOIS·Shodan·Maltego란 OSINT는 공개 source를 합법적으로 수집·검증해 의미 있는 intelligence로 만드는 과정이고, WHOIS/RDAP·Shodan·Maltego는 등록·노출 service·관계 분석을 돕는 도구다.

OSINT

Open Source Intelligence는 공개적으로 이용 가능한 source를 수집·검증·분석해 의사결정에 쓰는 정보다. 검색 결과를 그대로 사실로 보지 않고 source 신뢰성, timestamp, entity 일치, 법·privacy를 검토한다.

WHOIS·RDAP

domain registrar, registration date, name server, IP allocation 정보를 조회한다. privacy proxy와 registry policy로 개인 정보가 숨겨질 수 있다. RDAP은 structured HTTP 기반으로 WHOIS를 보완·대체한다.

Shodan

Internet에 노출된 service banner와 certificate를 검색하는 engine이다. 검색 결과는 scan 시점 snapshot이며 현재 exposure를 직접 확인하되 승인된 범위에서만 접근한다.

Maltego

domain, person, organization, infrastructure entity 사이 관계를 transform으로 시각화하는 OSINT 도구다. 자동 수집 결과에 같은 이름의 다른 사람이나 stale data가 섞일 수 있어 provenance를 확인한다.

OSINT/등록·DNS 기초 명령

```
# 공개 등록 정보와 DNS 확인
whois example.com
rdap example.com

dig example.com A
dig example.com MX
dig example.com NS
dig -x 192.0.2.10
```

Nmap·Port Scan·Service Detection

필기 확인 + 보충 해설

open이라는 결과가 어떤 packet 관찰에서 나온 것인지 읽는다

함께 찾을 말: Nmap, Port Scan, SYN Scan, Connect Scan, UDP Scan, Service Detection, OS Detection, Zenmap

Nmap·Port Scan·Service Detection란 port scan은 network endpoint의 응답을 관찰해 transport port 상태를 추정하고, Nmap은 host discovery, TCP/UDP scan, service·OS detection, script 기반 enumeration을 제공하는 도구다.

Port scan의 질문

"어떤 port가 open인가"는 해당 IP·transport에 probe를 보냈을 때 어떤 응답 상태가 관찰되는지를 뜻한다. open, closed, filtered, open|filtered는 scanner의 추론이며 firewall, packet loss, proxy가 영향을 준다.

TCP Connect와 SYN scan

connect scan은 OS의 connect()로 3-way handshake를 완료해 service log에 남기 쉽다. SYN scan은 SYN-ACK를 받으면 open으로 판단하고 connection을 완전히 성립시키지 않는 방식이다. 둘 다 대상에 traffic을 보내며 authorization과 rate control이 필요하다.

UDP scan

UDP는 handshake가 없어 응답 없음이 open 또는 filtered일 수 있다. ICMP Port Unreachable가 오면 closed로 추론한다. protocol-specific payload와 충분한 timeout이 필요해 느리다.

Nmap service/OS detection

port 번호만이 아니라 probe response로 application protocol과 version을 추정하고, TCP/IP stack response feature로 OS를 추정한다. middlebox와 banner masking 때문에 확정값이 아니므로 packet evidence와 server inventory를 대조한다.

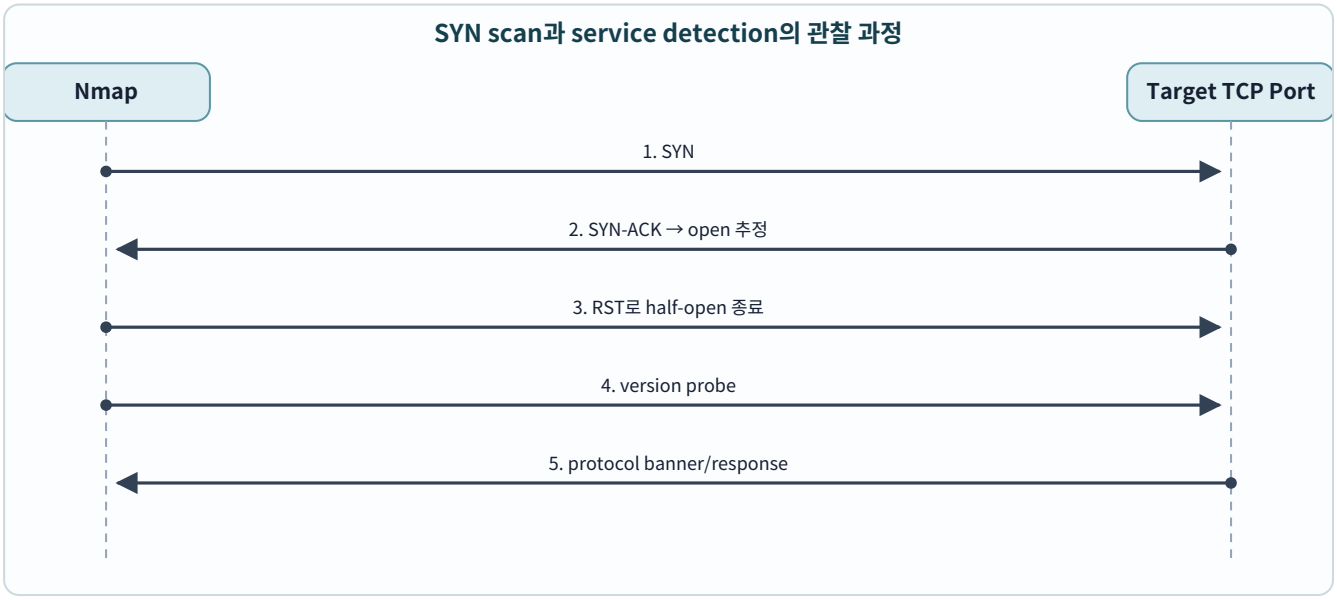


그림 32-2. “stealth scan”도 network IDS와 host firewall에 충분히 보일 수 있다.

안전한 범위의 Nmap 기본 예시

```
# 반드시 승인된 lab 대역에서만
nmap -sn 192.0.2.0/28
nmap -sT -sV --version-light 192.0.2.10
sudo nmap -sS -p 22,80,443 --reason 192.0.2.10
sudo nmap -sU -p 53,123,161 --reason 192.0.2.10
nmap -oA evidence/web01 192.0.2.10
```

Scan rate

`-T5`나 높은 `--min-rate`는 약한 장비·IPS·WAN을 압박하고 결과 정확도도 떨어뜨릴 수 있다. production 평가는 낮은 속도에서 시작하고 담당자와 monitoring을 공유한다.

NSE · DNS enumeration · Web discovery · Vulnerability Scanner

필기 확인 + 보충 해설

자동 도구 결과를 검증 가능한 가설로 취급한다

함께 찾을 말: NSE, Nmap Scripting Engine, DNSRecon, DNSenum, Nikto, Gobuster, ffuf, Nessus

NSE · DNS enumeration · Web discovery · Vulnerability Scanner란 NSE는 Nmap을 script로 확장하고, DNS/web enumeration 도구는 노출된 이름·path를 체계적으로 찾으며, vulnerability scanner는 알려진 weakness candidate를 자동 점검한다.

NSE

Nmap Scripting Engine은 Lua script로 discovery, version, safe vulnerability check, authentication test 등을 확장한다. category가 safe라고 해도 대상 영향이 0이라는 보장은 아니며 script source와 argument를 검토한다.

DNSRecon·DNSenum

DNS record, zone transfer 여부, common subdomain, reverse range 등을 enumeration한다. brute-force wordlist는 query 부하와 logging을 발생시킨다. DNSSEC 확인은 `dig +dnssec`와 chain validation tool을 사용할 수 있다.

Nikto·Gobuster·ffuf

Nikto는 web server의 위험한 file/configuration/signature를 점검하고, Gobuster와 ffuf는 wordlist로 path, virtual host, parameter candidate를 찾는다. 404 처리와 wildcard response를 calibrate하지 않으면 false positive가 많다.

Nessus

plugin 기반 vulnerability scanner로 service version, configuration, credentialed check를 통해 candidate finding을 만든다. credentialed scan은 patch와 local config를 더 정확히 보지만 강한 account와 agent impact를 관리해야 한다.

Enumeration 예시

```
# NSE script 목록/설명 확인 후 승인된 lab에서
nmap --script-help safe
nmap -sV --script http-title,http-headers 192.0.2.10

# Web content discovery: 낮은 thread에서 시작
ffuf -u http://192.0.2.10/FUZZ -w small-wordlist.txt -t 5 -mc all -fc 404
gobuster dir -u http://192.0.2.10/ -w small-wordlist.txt -t 5
```

33장. Packet 분석과 공격 개념: Sniffing·Spoofing·MITM·Injection

> packet tool을 안전하게 쓰려면 먼저 관찰, 위조, 중계, 주입을 구분해야 한다. Wireshark는 capture를 해석하는 도구이고, Scapy는 packet을 생성·수정할 수 있다. 기능이 강할수록 승인된 lab과 명확한 목적이 필요하다.

Sniffing·Spoofing·MITM

필기 확인 + 보충 해설

관찰 권한, 발신지 위조, 경로 장악을 분리한다

함께 찾을 말: Sniffing, Spoofing, Man in the Middle, MITM, ARP Spoofing, Session Sidejacking

Sniffing·Spoofing·MITM란 sniffing은 traffic을 관찰하고, spoofing은 origin field를 위조하며, MITM은 두 endpoint 사이에서 traffic을 중계·관찰·변조하는 위치를 획득하는 공격이다.

Sniffing

network traffic을 capture해 header와 payload를 관찰하는 행위다. 자기 host의 traffic, SPAN/TAP 복사본, wireless monitor mode 등 합법적 visibility가 필요하다. encrypted traffic도 IP/port/timing/certificate 같은 metadata는 보일 수 있다.

Spoofing

source IP, MAC, DNS answer, email sender처럼 identity·origin을 나타내는 field를 위조하는 행위다. field를 바꿀 수 있다는 것과 reply를 실제로 받아 session을 완성할 수 있다는 것은 다르다. anti-spoofing, authentication, cryptographic integrity가 필요하다.

Man-in-the-Middle

두 당사자 사이에 위치해 traffic을 relay하면서 읽거나 변경하는 공격 위치다. ARP spoofing, rogue AP, proxy, DNS manipulation 등으로 경로를 얻을 수 있지만 TLS certificate 검증과 end-to-end encryption이 content 변조를 막을 수 있다.

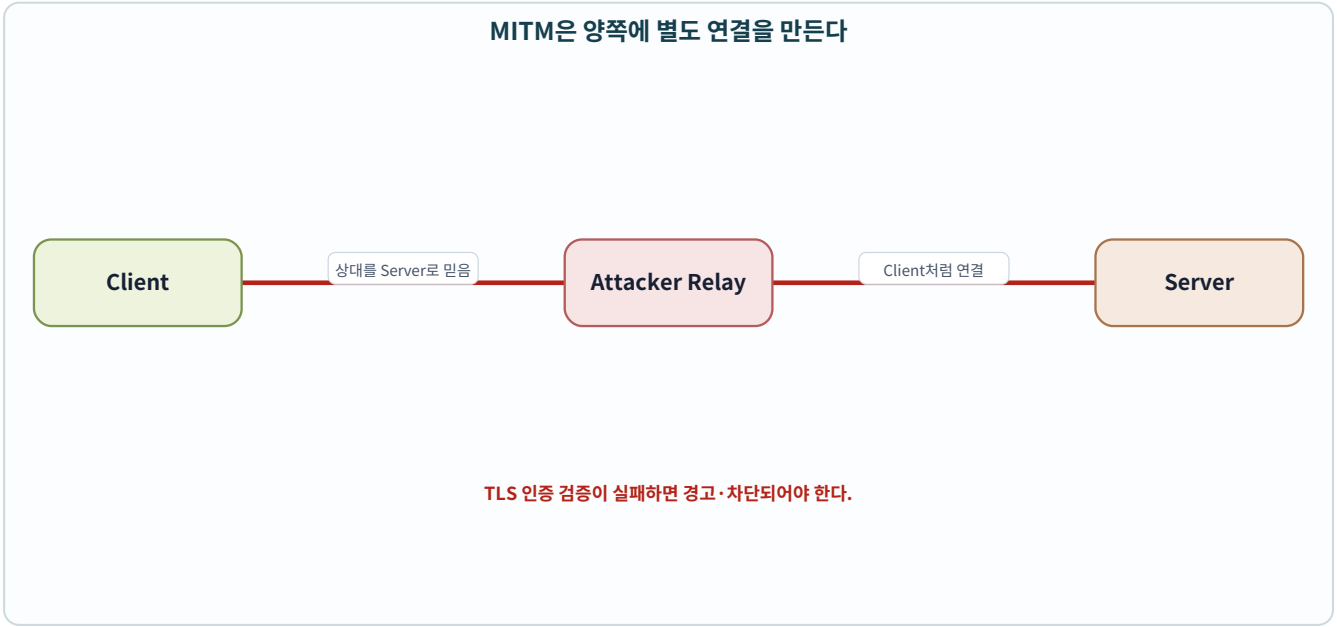


그림 33-1. 단순 packet capture와 traffic path를 실제로 가로채는 MITM은 다른 능력이다.

필기 확인 + 보충 해설

Packet Injection·Scapy·Fuzzing

패킷 생성 능력을 protocol 시험과 승인된 검증에 사용한다

함께 찾을 말: Packet Injection, Scapy, Packet Builder, Fuzzing

Packet Injection·Scapy·Fuzzing란 packet injection은 임의로 구성한 network PDU를 송신하는 행위이고, Scapy는 packet layer를 programmatically 생성·분석하는 framework이며, fuzzing은 변형 입력으로 오류를 찾는 시험 기법이다.

Packet injection

직접 구성한 frame/packet을 interface로 송신하거나 기존 stream에 조건을 맞춘 packet을 삽입하는 행위다. sequence/ACK, checksum, MAC, route를 맞춰야 하며 switch/router/host가 검증할 수 있다. 무선 deauthentication이나 TCP reset injection 등은 서비스에 영향을 줄 수 있어 격리 lab에서만 수행한다.

Scapy

Python으로 packet layer를 조합하고 send/sniff/pcap 처리하는 framework다. protocol 학습, test fixture, fuzzing에 유용하다. root/capability가 필요한 raw socket과 잘못된 destination으로 traffic이 나가지 않도록 namespace·isolated network를 사용한다.

Scapy packet builder

```
# 로컬 격리 lab에서 packet 구조만 만들고 출력하는 예
from scapy.all import Ether, IP, TCP, hexdump

pkt = (Ether(dst="02:00:00:00:00:02") /
       IP(src="192.0.2.10", dst="192.0.2.20") /
       TCP(sport=40000, dport=8080, flags="S"))

pkt.show()
hexdump(pkt)
# sendp(pkt, iface="lab0")는 소유한 격리 interface에서만 사용
```

Fuzzing

입력 format을 변형·생성해 crash, hang, memory corruption, logic error를 찾는 test 방식이다. dumb random mutation보다 grammar/protocol-aware, coverage-guided fuzzing이 깊은 code path를 탐색할 수 있다. target을 container/VM에 격리하고 crash input과 version을 보존한다.

Packet builder와 checksum

field를 직접 만들면 length와 checksum이 자동 계산되거나 명시값으로 고정될 수 있다. capture에서 offload와 fragmentation을 고려해 실제 wire bytes를 검증한다.

LAND·Teardrop·DDoS·Brute Force

필기 확인 + 보충 해설

packet 모양과 자원 고갈 메커니즘으로 공격을 분류한다

함께 찾을 말: LAND Attack, Teardrop, Fragmentation Attack, DDoS, RDoS, Brute Force

LAND·Teardrop·DDoS·Brute Force란 LAND와 Teardrop은 비정상 주소·fragment 처리 취약점을 악용한 역사적 DoS이고, DDoS는 분산 자원 고갈, brute force는 반복 credential 추측 공격이다.

LAND attack

source와 destination IP/port를 피해 host 자신으로 조작한 packet으로 오래된 TCP/IP stack의 비정상 처리와 resource loop를 유발하던 역사적 DoS 기법이다. 현대 stack은 대체로 방어하지만 spoofed packet filtering과 patch가 중요하다.

Teardrop과 fragmentation

서로 겹치거나 비정상 fragment offset/length를 보내 재조립 취약점을 악용한 역사적 DoS다. 현대 OS는 validation과 reassembly policy로 방어한다. IDS와 endpoint의 overlap 해석 차이는 evasion 연구에서 중요하다.

DDoS와 RDoS

DDoS는 여러 source에서 traffic이나 application request를 보내 자원을 고갈시키는 공격이다. RDoS는 ransom demand와 DDoS 위협/공격을 결합한다. rate limiting, upstream scrubbing, anycast/CDN, capacity, application cache, incident contact를 준비한다.

Brute Force

credential 후보를 반복 시험하는 공격이다. online brute force는 rate limit·lockout·MFA·risk-based authentication으로, offline hash cracking은 느린 salted KDF와 강한 password로 방어한다. account lockout은 DoS로 악용될 수 있어 progressive delay와 alert를 설계한다.

역사적 공격을 학습하는 이유

LAND·Teardrop command를 재현하는 것이 목적이 아니라, parser가 비정상 header와 fragment를 어떻게 검증해야 하는지, patch와 normalization이 왜 필요한지 이해하는 데 목적이 있다.

Wireshark·Npcap·tcpdump·PCAP 분석

필기 확인 + 보충 해설

packet 한 장보다 대화와 전후 dependency를 읽는다

함께 찾을 말: Wireshark, Npcap, tcpdump, Capture Filter, Display Filter, PCAP

> **Wireshark·Npcap·tcpdump·PCAP 분석란** Wireshark와 tcpdump는 packet capture와 protocol field 분석을 제공하고, Npcap은 Windows capture driver이며, 정확한 분석은 capture 위치·filter·offload·시간을 함께 기록한다.

Wireshark

libpcap/Npcap 등으로 capture하거나 pcap을 열어 protocol dissector로 field를 해석한다. display filter는 이미 capture한 packet 표시를, capture filter(BPF syntax)는 수집 단계에서 packet을 줄인다. 둘의 문법이 다르다.

Npcap

Windows에서 packet capture와 injection interface를 제공하는 driver/library로 Wireshark, Nmap 등이 사용한다. installation mode, loopback capture, admin 권한, driver signature를 관리한다.

tcpdump

CLI packet capture tool로 remote server와 low-overhead 증거 수집에 적합하다. `-s 0`은 full snaplen, `-n`은 name resolution 비활성화, `-w`는 pcap 저장이다. terminal에 payload를 출력하면 민감 정보가 노출될 수 있다.

Capture와 display filter 예시

```
# 승인된 host에서 특정 service flow 캡처
sudo tcpdump -ni eth0 -s 0 -w web443.pcap 'host 192.0.2.10 and tcp port 443'

# Wireshark display filter 예
tcp.flags.syn == 1 && tcp.flags.ack == 0
ip.addr == 192.0.2.10 && tcp.port == 443
dns.flags.rcode != 0
```

분석 순서

1. capture 위치와 시간·offload 상태를 기록한다.
2. conversation/endpoint 통계로 범위를 좁힌다.
3. ARP/DNS/TCP handshake처럼 선행 dependency를 확인한다.
4. retransmission·RST·ICMP 오류와 application response를 시간순으로 본다.
5. server log와 같은 timestamp로 교차 검증한다.

34장. HTTP · Cookie · Session · Same-Origin · CORS: 웹 보안의 기반

> SQL injection과 XSS를 payload 문자열로 외우기 전에 browser와 server가 request, cookie, origin, HTML/JavaScript를 어떻게 처리하는지 알아야 한다. 취약점은 이 신뢰 경계에서 data와 code, identity와 intent를 혼동할 때 생긴다.

HTTP · Method · Status · Header

필기 확인 + 보충 해설

웹 취약점이 발생하는 message 경계

함께 찾을 말: HTTP, HTTPS, Request, Response, Method, Status Code, Header, X-Header

HTTP · Method · Status · Header란 HTTP는 client와 server가 request와 response message를 교환하는 application protocol이고, method는 요청 의도, status는 처리 결과 범주, header는 message metadata를 표현한다.

HTTP request/response

request는 method, target, version, header, optional body로 구성되고 response는 status, header, body를 가진다. HTTP는 application message protocol이며 TCP/TLS/QUIC 위에서 전달될 수 있다.

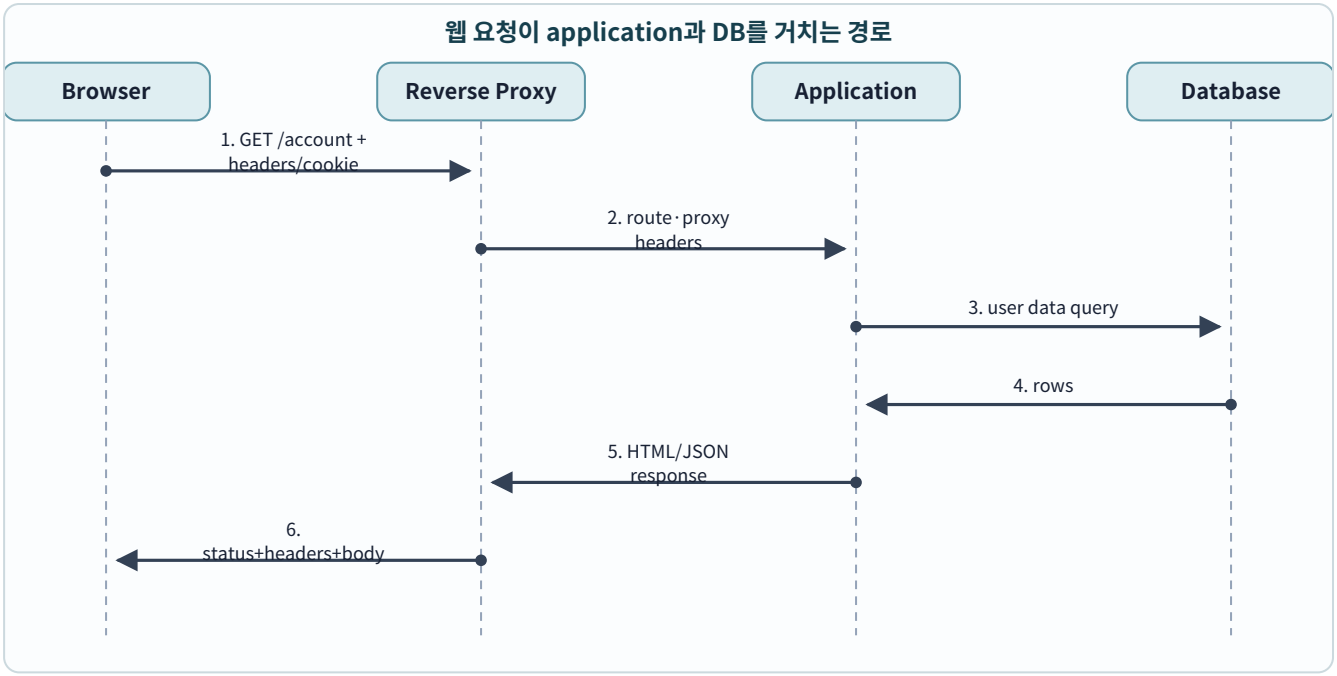


그림 34-1. 각 경계에서 입력 검증, identity 전달, output encoding, timeout, logging이 필요하다.

Method와 status

GET은 safe/idempotent 의미를 목표로 하고 조회에, POST는 상태 변경에 흔히 쓰지만 실제 보안은 method 이름만 믿지 않는다. 2xx success, 3xx redirect, 4xx client/request 문제, 5xx server error 범주이며 status와 body가 일치하는지 본다.

Header

Host, Content-Type, Authorization, Cookie, Origin, Referer, Cache-Control, X-Forwarded-For 등 message metadata다. X- 접두사는 historical custom header 관례였고 표준화되지 않은 이름을 의미했으나 현재는 반드시 X-를 쓰지 않는다. proxy header는 trusted proxy에서만 신뢰한다.

Cookie·Session·Session Hijacking

필기 확인 + 보충 해설

로그인 상태를 유지하는 값이 곧 인증 credential이 된다

함께 찾을 말: Cookie, Session, Session ID, Session Hijacking, Sidejacking, HttpOnly, Secure, SameSite, Zero-click, One-click

Cookie·Session·Session Hijacking란 cookie는 browser가 조건에 따라 request에 자동 첨부하는 client 저장 값이고, session은 server가 여러 request를 같은 identity·state로 연결하는 구조이며, hijacking은 session credential을 탈취해 identity를 도용하는 공격이다.

Cookie

server가 Set-Cookie로 browser에 저장을 요청하고, browser가 domain/path/SameSite/Secure 조건에 맞는 후속 request에 Cookie header로 자동 첨부하는 작은 값이다. session ID, preference에 사용한다. HttpOnly는 JavaScript read를 제한하고 Secure는 HTTPS에서만 전송한다.

Session

server가 login 상태와 user context를 유지하는 논리적 대화다. 흔히 random session ID만 cookie에 두고 실제 state는 server-side store에 저장한다. token/JWT 기반에서는 state 일부가 client token에 포함될 수 있지만 revocation과 key 관리가 필요하다.

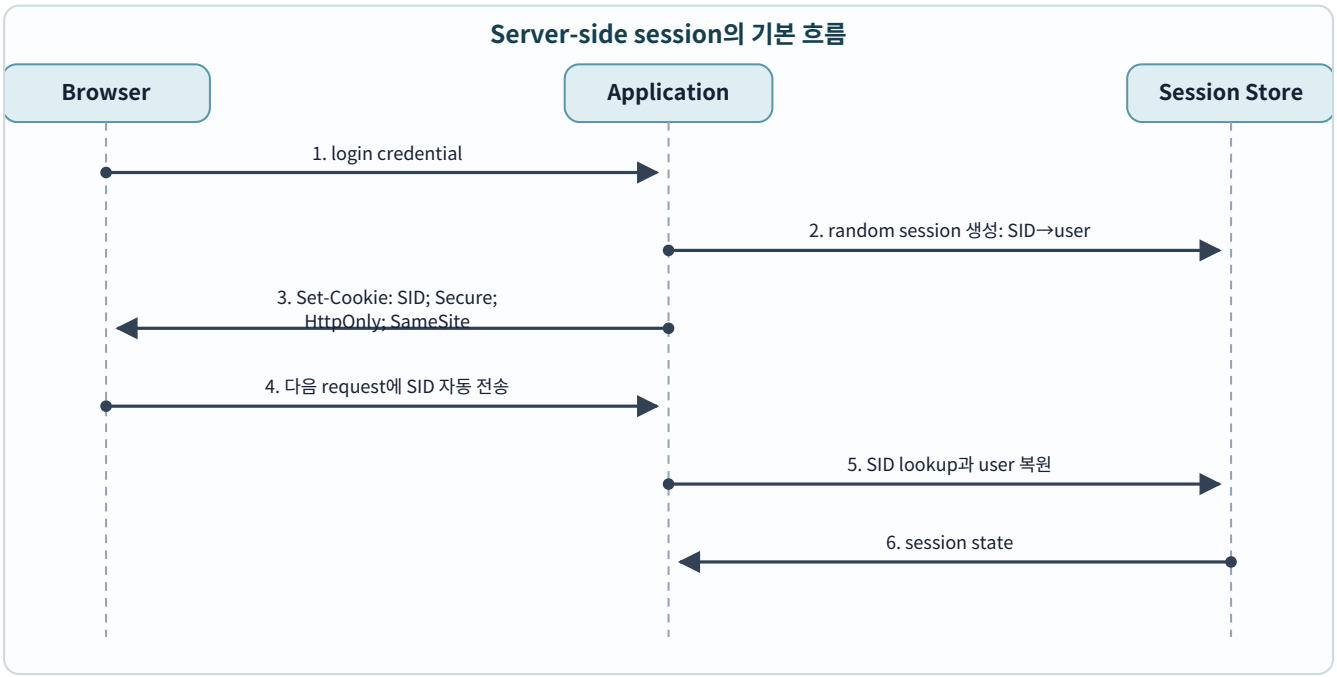


그림 34-2. session ID를 아는 주체를 로그인 user로 간주하므로 entropy, transport, rotation, expiry가 중요하다.

Session hijacking과 sidejacking

session ID를 탈취·고정·예측해 user로 가장하는 공격이다. sidejacking은 평문 wireless/HTTP traffic 등에서 cookie를 훔치는 문맥에 쓰인다. TLS, HttpOnly, Secure, SameSite, login 시 rotation, 짧은 expiry, device/risk signal로 완화한다.

One-click와 zero-click

one-click exploit은 user가 link/file을 한 번 열거나 interaction해야 하고, zero-click은 메시지 수신·preview 같은 자동 처리만으로 exploit되는 공격을 뜻한다. 공식 taxonomy라기보다 interaction 요구 정도를 나타내는 표현이다.

Same-Origin Policy · CORS · Origin

필기 확인 + 보충 해설

Browser 정책과 Server 접근통제를 혼동하지 않는다

함께 찾을 말: Same-Origin Policy, SOP, CORS, Origin, Preflight, Access-Control-Allow-Origin

Same-Origin Policy · CORS · Origin란 origin은 scheme · host · port 조합이고, Same-Origin Policy는 다른 origin script의 data read를 제한하며, CORS는 server가 선택적으로 cross-origin read를 허용하는 response protocol이다.

Origin

scheme, host, port의 세 요소가 같은지로 정의된다. `https://app.example:443`와 `http://app.example:80`는 scheme이 달라 다른 origin이다. path는 origin 구성 요소가 아니다.

Same-Origin Policy

악성 site의 script가 다른 origin의 response data를 임의로 읽지 못하게 browser가 제한하는 핵심 정책이다. cross-origin request 자체가 항상 막히는 것은 아니다. form, image, script load처럼 보내거나 embed할 수 있지만 response read가 제한되는 경우가 있다.

CORS

server가 Access-Control-Allow-Origin 등 response header로 특정 cross-origin JavaScript read를 허용하는 mechanism이다. browser enforcement이며 server-to-server client나 curl을 막는 firewall이 아니다. credential을 허용하면서 origin을 무분별하게 reflect하면 data exposure가 생긴다.

Preflight

복잡한 method/header request 전에 browser가 OPTIONS로 server 허용 정책을 묻는 절차다. simple request는 preflight 없이 전송될 수 있어 CORS가 CSRF 방어와 동일하지 않다.

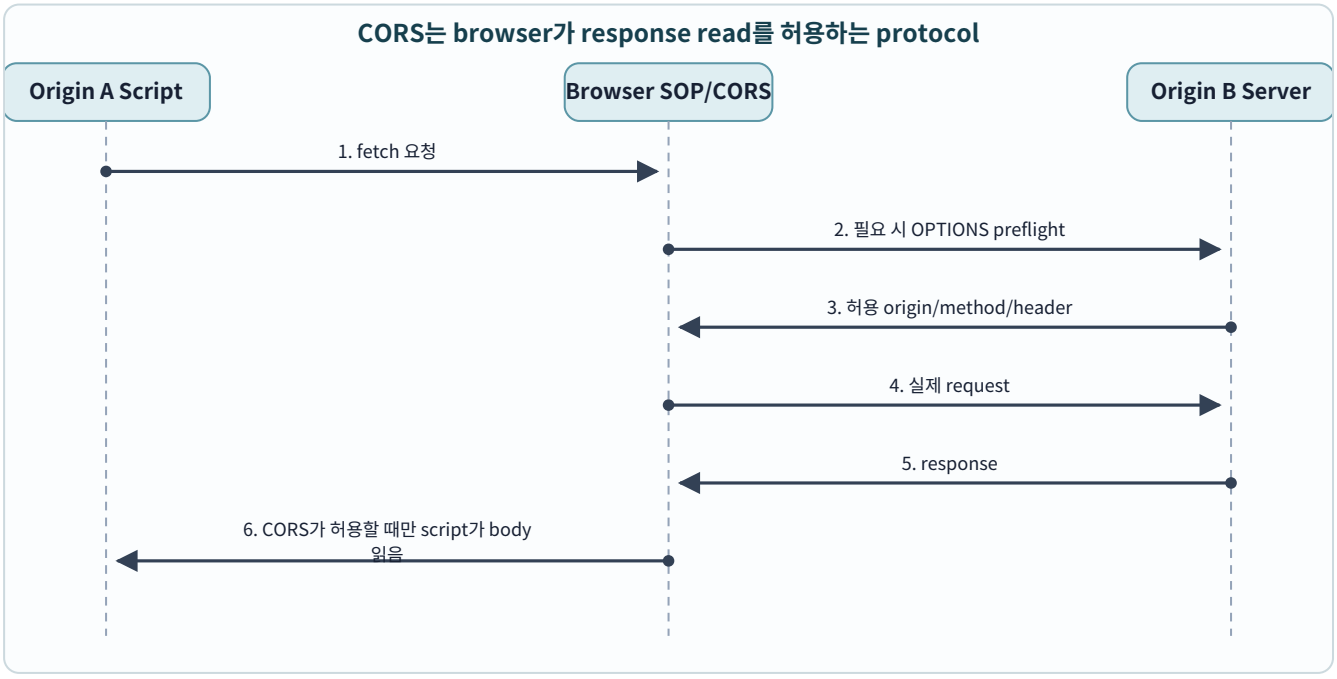


그림 34-3. server는 여전히 authentication, authorization, CSRF 방어를 수행해야 한다.

HTML Context·iframe·URL Encoding·CSP

필기 확인 + 보충 해설

같은 문자열도 어느 parser가 읽는지에 따라 code가 될 수 있다

함께 찾을 말: DOM, JavaScript Context, iframe, URL Encoding, Percent Encoding, Content Security Policy, CSP, Clickjacking

HTML Context·iframe·URL Encoding·CSP란 browser는 response를 HTML·attribute·JavaScript·URL context로 해석하며, iframe은 document를 embed하고, URL encoding은 byte 표현 규칙, CSP는 허용 resource와 script 실행을 제한하는 browser policy다.

HTML·DOM·JavaScript context

server response의 text가 HTML parser, attribute, JavaScript string, CSS, URL 중 어느 context에 삽입되는지에 따라 필요한 encoding이 다르다. <를 HTML entity로 바꾸는 것만으로 JavaScript string context가 안전해지는 것은 아니다.

iframe

한 document 안에 다른 page를 embed하는 element다. same-origin이면 parent/child script가 DOM에 접근할 수 있고, cross-origin이면 SOP가 제한한다. sandbox attribute와 CSP frame-ancestors로 기능·embedding을 제한할 수 있다. clickjacking은 투명 iframe으로 user click을 유도한다.

URL encoding

URL의 reserved character와 non-ASCII byte를 percent-encoding한다. encoding은 보안 검증이 아니며 여러 번 decode되는 경로에서 filter bypass가 생길 수 있다. canonicalization 후 allowlist와 context-aware output encoding을 한다.

Content Security Policy

CSP는 script, style, frame, connection 등의 resource source를 response header로 제한해 XSS 영향과 data exfiltration을 줄이는 defense-in-depth다. unsafe-inline/unsafe-eval이나 broad wildcard는 효과를 약화한다.

35장. SQL Injection·XSS·CSRF·접근통제: 취약점이 생기는 메커니즘

> 웹 취약점의 핵심은 공격 문자열을 외우는 것이 아니라, 신뢰할 수 없는 data가 어떤 interpreter의 code 영역에 들어가거나, server가 identity·권한·요청 의도를 잘못 신뢰하는지를 찾는 것이다.

SQL Injection · Blind SQL Injection

필기 확인 + 보충 해설

입력값 검사가 아니라 code와 data의 구조적 분리가 핵심

함께 찾을 말: SQLi, SQL Injection, Blind SQL Injection, Prepared Statement, Parameterized Query

SQL Injection · Blind SQL Injection란 SQL injection은 신뢰할 수 없는 입력이 SQL 문법으로 해석되어 query 구조를 바꾸는 취약점이고, blind 방식은 response 차이·시간 등 간접 신호로 결과를 추론한다.

SQL과 data 경계가 무너지는 순간

application이 `"SELECT ... WHERE name = '" + input + "'"`처럼 문자열 결합으로 query를 만들면 input의 quote와 operator를 DB parser가 SQL syntax로 해석할 수 있다. attacker data가 query structure를 바꾸는 것이 injection이다.

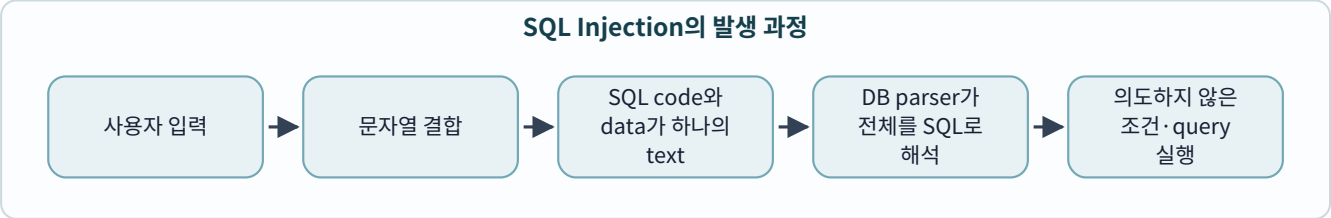


그림 35-1. prepared statement는 SQL 구조를 먼저 compile하고 input을 typed parameter data로 전달해 경계를 유지한다.

Blind SQL Injection

query result가 화면에 직접 보이지 않아도 response의 참/거짓 차이, error, 시간 지연 같은 side channel로 data를 한 bit씩 추론하는 방식이다. “화면에 DB 내용이 안 나오므로 안전”하지 않다.

방어

- parameterized query/prepared statement를 모든 입력에 사용한다.
- table/column 이름처럼 parameter화할 수 없는 identifier는 allowlist mapping한다.
- DB runtime account를 최소 privilege로 제한한다.
- error detail을 client에 노출하지 않고 server log에 남긴다.
- input validation은 business rule용 보조 통제이며 escaping만으로 모든 DB/context를 해결하지 않는다.

SQL code와 parameter data 분리

```
# 취약한 예: 문자열 결합
sql = "SELECT * FROM users WHERE username = '" + username + "'"

# 안전한 개념 예: driver parameter binding
cursor.execute(
    "SELECT id, password_hash FROM users WHERE username = %s",
    (username,),
)
```

XSS

필기 확인 + 보충 해설

문자열이 browser parser에서 data가 아니라 code가 되는 순간

함께 찾을 말: Cross-Site Scripting, Stored XSS, Reflected XSS, DOM XSS, Output Encoding, Sanitization

XSS란 XSS는 공격자 data가 web page의 실행 가능한 HTML/JavaScript context에 들어가 victim browser가 해당 site origin 권한으로 code를 실행하는 취약점이다.

XSS가 실행되는 위치

server 또는 client script가 공격자 입력을 HTML/JavaScript context에 안전하게 encoding하지 않아 victim browser가 이를 code로 실행한다. 공격은 server에서 실행되는 것이 아니라 해당 origin 권한을 가진 browser script로 실행된다.

Stored · Reflected · DOM-based

Stored XSS

악성 input이 DB/post/profile에 저장되어 여러 user response에 포함.

Reflected XSS

request parameter가 즉시 response에 반사.

DOM-based XSS

client JavaScript가 location/hash/message 등을 unsafe sink(innerHTML, eval 등)에 넣어 browser 안에서 발생.

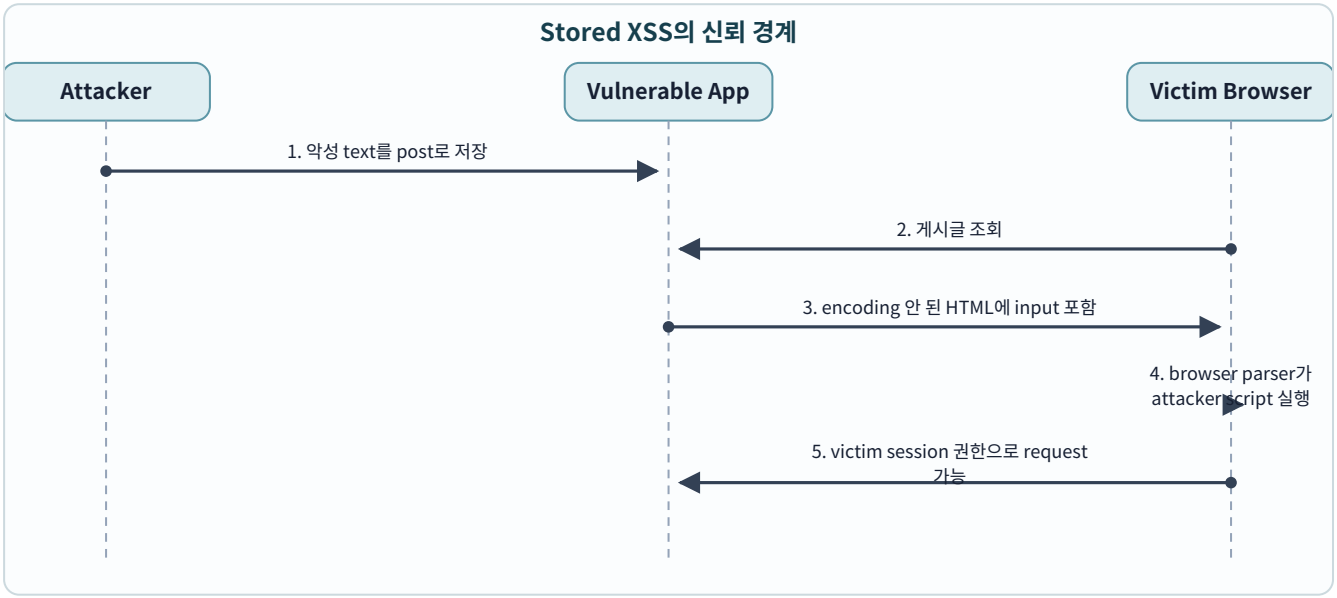


그림 35-2. HttpOnly는 cookie read를 줄이지만 XSS가 같은 origin에서 request를 보내는 능력까지 없애지는 못한다.

Context-aware output encoding

HTML text, attribute, JavaScript, CSS, URL은 각각 다른 encoder가 필요하다. 가능하면 template engine의 auto-escaping을 유지하고 DOM에는 textContent, 안전한 attribute API를 사용한다. rich HTML이 필요하면 검증된 sanitizer를 allowlist로 적용한다.

CSP와 Trusted Types

CSP nonce/hash와 Trusted Types는 inline script와 dangerous DOM sink 사용을 제한해 XSS exploitability를 줄이는 보조 통제다. output encoding과 safe API를 대체하지 않는다.

CSRF

필기 확인 + 보충 해설

신원이 맞다는 사실과 사용자가 그 작업을 의도했다는 사실은 다르다

함께 찾을 말: Cross-Site Request Forgery, CSRF Token, SameSite, Origin Check

CSRF란 CSRF는 victim browser가 target site credential을 자동 첨부한다는 점을 악용해 victim이 의도하지 않은 state-changing request를 보내게 하는 공격이다.

CSRF가 가능한 조건

browser가 target site cookie를 request에 자동 첨부하고, server가 그 cookie의 로그인 상태만 보고 요청이 user 의도라고 믿을 때 attacker site가 form/image/fetch 등의 방식으로 상태 변경 request를 유도할 수 있다. attacker가 response를 읽지 못해도 요청 수행이 목적이면 성공할 수 있다.

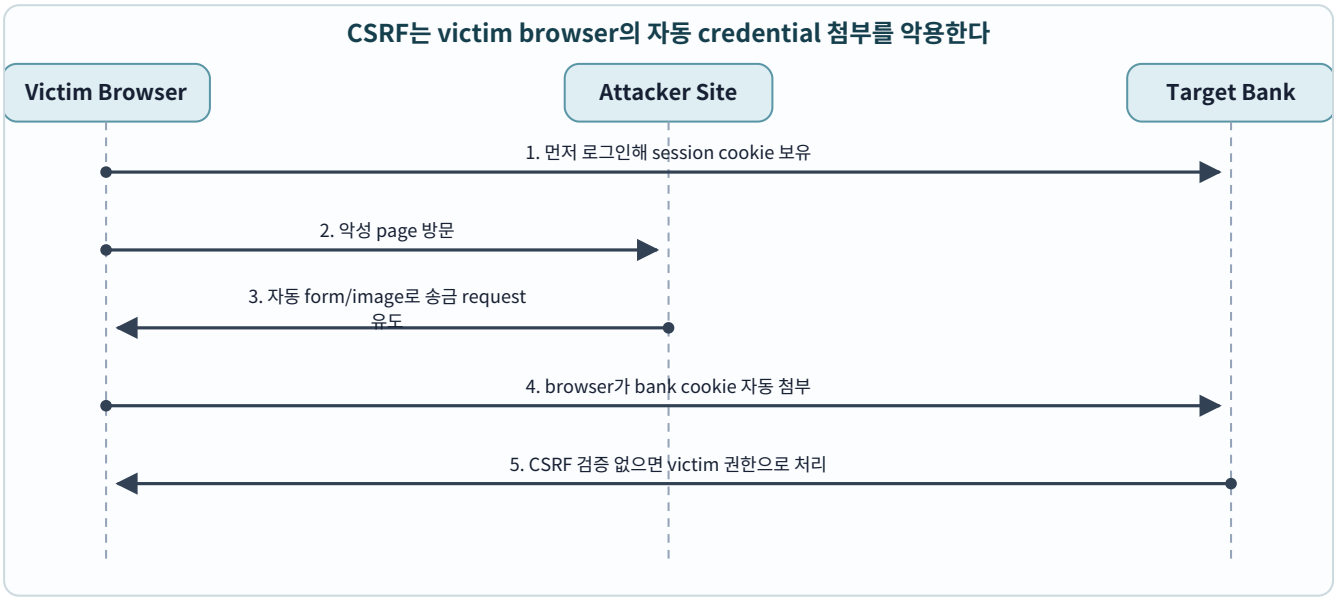


그림 35-3. XSS는 target origin에서 code를 실행하고, CSRF는 cross-site에서 요청 의도를 위조한다. 서로 결합될 수 있지만 같은 취약점은 아니다.

방어

- 예측 불가능하고 session에 결합된 CSRF token을 state-changing request마다 검증한다.
- SameSite cookie를 Lax/Strict로 설정하되 예외와 browser behavior를 이해한다.
- Origin/Referer를 보조 검증하고 없을 때 정책을 정한다.
- 중요 작업에 password/MFA 재인증과 transaction confirmation을 적용한다.
- GET에 상태 변경을 배치하지 않는다.

CORS는 CSRF 방어가 아니다

simple cross-origin form request는 preflight 없이 전송될 수 있고, CORS는 주로 attacker script가 response를 읽을 수 있는지 제어한다. custom header+preflight를 CSRF 통제로 활용할 수 있어도 server-side token과 SameSite 설계를 대신하지 않는다.

Broken Access Control·IDOR/BOLA

필기 확인 + 보충 해설

로그인 여부가 아니라 "이 사용자가 이 객체에 이 작업을 할 수 있는가"

함께 찾을 말: Broken Access Control, IDOR, BOLA, Horizontal Privilege Escalation, Vertical Privilege Escalation

Broken Access Control·IDOR/BOLA란 broken access control은 server가 identity의 resource·action 권한을 올바르게 검사하지 않는 취약점이고, IDOR/BOLA는 object identifier를 바꿔 다른 object에 접근하는 대표 형태다.

Broken Access Control

authenticated user가 허용되지 않은 object·function·role에 접근할 수 있는 상태다. URL을 숨기거나 button을 비활성화하는 client-side 조치는 server authorization이 아니다.

IDOR/BOLA

request의 object ID를 바꾸었을 때 server가 해당 object의 owner/tenant/role을 확인하지 않아 다른 user data를 읽거나 수정하는 취약점이다. API에서는 BOLA라는 용어가 널리 쓰인다.

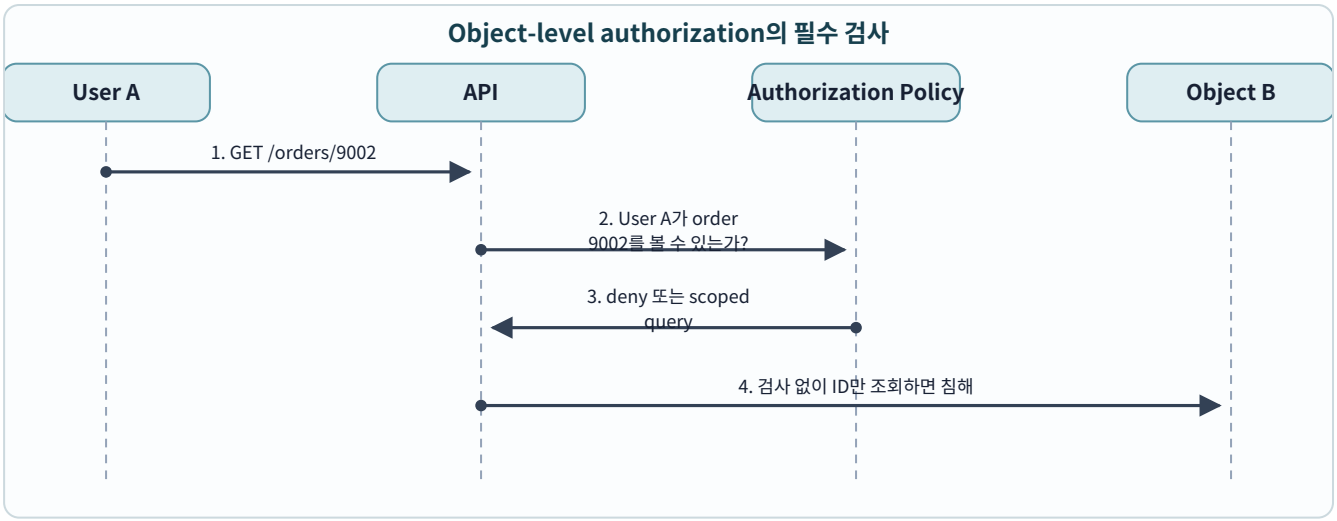


그림 35-4. ID가 random UUID여도 guess가 어려울 뿐, authorization을 대체하지 않는다.

Horizontal과 Vertical

horizontal privilege escalation은 같은 role의 다른 user resource에 접근, vertical은 일반 user가 admin 기능 등 더 높은 role action을 수행하는 것이다.

방어

- server-side에서 매 request의 subject, action, object, context를 검사한다.
- DB query 자체를 tenant/user scope로 제한한다.
- deny by default와 centralized policy를 사용한다.

- role matrix와 negative test를 자동화한다.
- authorization failure를 audit하되 object 존재 여부를 과도하게 노출하지 않는다.

Command Injection·File Upload·SSRF·보안 로깅

필기 확인 + 보충 해설

각기 다른 interpreter와 trust boundary가 무너지는 방식

함께 찾을 말: Command Injection, File Upload Vulnerability, SSRF, Security Logging and Monitoring Failures

Command Injection·File Upload·SSRF·보안 로깅란 command injection은 input이 OS command 문법이 되는 취약점, unsafe upload는 공격 file이 실행·노출되는 문제, SSRF는 server의 network 권한을 악용한 request 위조, logging failure는 사건 evidence와 detection 부족이다.

Command Injection

application input이 shell command 문자열에 결합되어 shell metacharacter와 추가 command가 실행되는 취약점이다. shell을 거치지 않는 process API와 argument list, allowlist, least privilege를 사용한다.

File Upload

extension과 Content-Type만 믿으면 script, polyglot, path traversal, oversized file이 들어올 수 있다. random server-side name, content validation, size limit, non-executable storage, malware scan, separate origin download를 적용한다.

SSRF

server가 user-controlled URL로 request를 보내게 해 internal service, cloud metadata, localhost에 접근시키는 취약점이다. scheme/host/IP allowlist, DNS rebinding 대응, redirect 재검증, egress firewall을 사용한다.

Security Logging Failure

authentication failure, authorization denial, admin action, input validation failure가 충분한 context와 함께 기록·경보되지 않아 공격을 발견·조사하지 못하는 상태다. secret와 password를 log에 남기는 과잉 logging도 위험하다.

36장. Kali 도구와 취약 실습 플랫폼: 프로그램 이름을 역할별로 정리하기

> 도구 이름을 길게 외우는 대신 “발견, 검증, credential, web, wireless, post-exploitation, forensic” 역할로 분류하고, 어떤 입력과 권한이 필요한지 본다. 사용 명령은 소유한 격리 lab에 한정한다.

Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

필기 확인 + 보충 해설

취약 VM는 Internet에 노출하지 않고 snapshot으로 되돌린다

함께 찾을 말: Kali Linux, Metasploit, Metasploitable, DVWA, WebGoat, bee-box, bWAPP, Docker Lab

> **Kali Linux · Metasploitable · DVWA · WebGoat · bee-box란** Kali는 security testing 도구를 모은 Linux distribution이고, Metasploitable · DVWA · WebGoat · bee-box는 취약점을 안전하게 학습하도록 의도적으로 약점을 포함한 격리 실습 환경이다.

Kali Linux

penetration testing과 digital forensics 도구를 package와 menu로 제공하는 Debian 계열 distribution이다. Kali 자체가 "해킹 기능"을 자동 제공하는 것이 아니라 일반 Linux 위에 도구와 wordlist, 설정을 모은 환경이다. daily driver보다 disposable VM와 snapshot을 쓰는 경우가 많다.

취약 실습 플랫폼

플랫폼	학습 영역	운영 주의
Metasploitable	의도적으로 취약한 OS/service	외부 network에 연결 금지
DVWA	PHP 기반 web 취약점	security level과 DB 초기화
WebGoat	lesson 기반 web security	Java app, 최신 lesson 차이
bee-box/bWAPP	다양한 web bug	격리 VM
Security Shepherd 등	CTF/lesson	배포 version 확인

Docker 실습

취약 application을 container로 빠르게 배포할 수 있지만 host port 공개, privileged mode, volume mount, Docker socket 노출을 주의한다. container reset은 쉽지만 host kernel을 공유하므로 untrusted exploit lab의 완전한 boundary로 단독 신뢰하지 않는다.

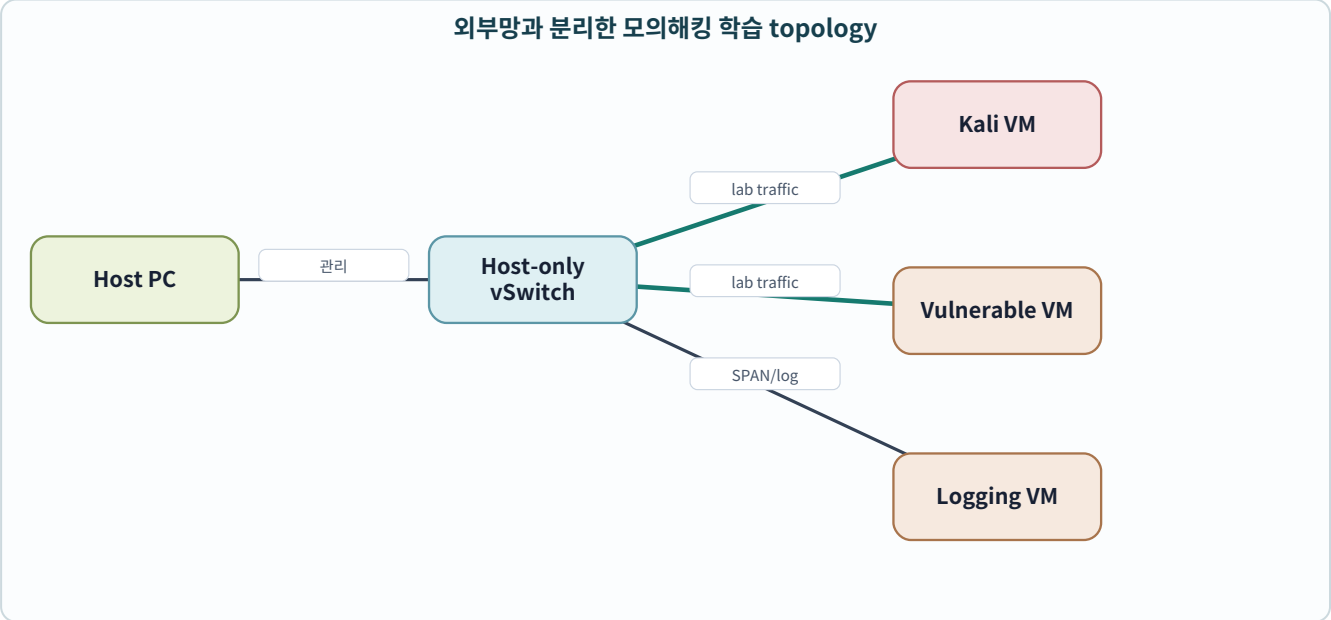


그림 36-1. NAT Internet update path와 vulnerable lab path를 분리하고 forwarding을 제한한다.

필기 확인 + 보충 해설

Burp Suite · sqlmap · Hydra · John · Hashcat

자동화 능력보다 시험 범위와 request 영향이 먼저

함께 찾을 말: Burp Suite, sqlmap, Hydra, John the Ripper, Hashcat, Aircrack-ng, WPScan, BeEF, SET

Burp Suite · sqlmap · Hydra · John · Hashcat란 Burp는 web request를 관찰·재전송하는 proxy, sqlmap은 SQLi 검증 자동화, Hydra는 online authentication test, John/Hashcat은 offline password hash audit 도구다.

도구	역할	핵심 주의
Burp Suite	HTTP proxy, repeater, scanner	proxy certificate와 scope 제한
sqlmap	SQLi 자동 검증	DB 부하·data extraction 최소화
Hydra	online login credential test	rate · lockout · 승인
John/Hashcat	offline password hash audit	허가된 hash와 GPU 자원
Aircrack-ng	802.11 capture/audit	전파 범위와 소유 AP
WPScan	WordPress enumeration	API token · rate
BeEF	browser security lab	hook를 외부에 노출 금지
SET	social engineering toolkit	사람 대상은 별도 명시 승인

Burp Suite

browser와 web server 사이의 intercepting proxy로 request를 capture·modify하고 Repeater에서 반복 검증하며, Proxy history와 Logger로 evidence를 남긴다. HTTPS를 보기 위해 lab browser에 Burp CA를 설치하므로 일반 browser profile과 분리한다.

sqlmap

parameter의 response 차이와 DB dialect를 시험해 SQL injection을 자동 검증한다. 기본 탐지조차 여러 request를 보내므로 production scope, risk/level, timeout을 낮게 시작하고 DB dump·OS command 기능은 별도 승인 없이 사용하지 않는다.

Hydra와 password audit

online authentication에 credential candidate를 시험한다. account lockout과 service outage를 일으킬 수 있다. John/Hashcat은 합법적으로 획득한 password hash를 offline 분석해 password policy와 hash 비용을 평가한다. 결과 password는 최소 노출하고 즉시 reset한다.

Post-exploitation·LinPEAS·Mimikatz·Responder·NetExec·Chisel·Autopsy

필기 확인 + 보충 해설

강한 도구는 최소 증명·증거 보호·cleanup까지 scope에 포함한다

함께 찾을 말: Post Exploitation, LinPEAS, Mimikatz, Responder, NetExec, Chisel, Autopsy, PyInstaller

Post-exploitation·LinPEAS·Mimikatz·Responder·NetExec·Chisel·Autopsy란 post-exploitation 도구는 초기 access 이후 privilege·credential·lateral path·evidence를 평가하고, forensic 도구는 침해 흔적과 disk data를 보존·분석한다.

도구	범주	설명
LinPEAS	Linux local enumeration	permission, service, secret, misconfiguration candidate 수집
Mimikatz	Windows credential research	LSASS/credential material 접근; lab와 승인 필수
Responder	LLMNR/NBT-NS/mDNS poisoning lab	Windows name resolution 신뢰 악용
NetExec	Windows/AD remote protocol assessment	SMB/WinRM 등 credential·configuration 검증
Chisel	HTTP 기반 TCP tunnel	segmentation 우회 위험; lab only
Autopsy	Digital forensics	disk image, file system, timeline 분석
PyInstaller	Python packaging	script와 interpreter dependency를 executable bundle로 묶음

Post-exploitation이라는 말

초기 access 후 privilege, credential, persistence, lateral movement, data impact를 평가하는 단계다. 실제 환경에서는 최소한의 proof만 수행하고 secret dumping·persistence 설치·tunnel은 별도 승인한다.

LinPEAS

host configuration을 빠르게 enumeration해 writable path, sudo, SUID, service secret candidate를 표시하는 script다. 출력은 취약점 확정이 아니며 민감 정보가 많아 evidence를 암호화·삭제한다.

Mimikatz와 Responder

Mimikatz는 Windows authentication credential material 연구 도구로 방어 시험에 쓰이지만 credential theft 능력이 매우 강하다. Responder는 local name resolution fallback과 NTLM authentication 유도를 lab에서 검증한다. SMB signing, LLMNR/NBT-NS disable, EPA, credential hygiene가 관련 방어다.

Chisel

HTTP/WebSocket 위에 TCP tunnel을 구성할 수 있어 제한된 network에서 pivot에 사용될 수 있다. 승인 없는 tunnel은 security control 우회이므로 실제 assessment에서는 목적·destination·종료와 cleanup을 명시한다.

도구가 결과를 "증명"하지 않는다

LinPEAS의 빨간색, Nessus severity, Metasploit module 성공 메시지는 해석이 필요한 evidence다. version, privilege, network path, business impact를 수동 검증한다.

Metasploit·Exploit·Payload·Session·Reverse Shell

필기 확인 + 보충 해설

취약점 검증과 원격 제어 channel을 별도 단계로 본다

함께 찾을 말: Metasploit Framework, Exploit, Payload, Session, Reverse Shell

Metasploit·Exploit·Payload·Session·Reverse Shell란 Metasploit은 exploit과 payload·session을 관리하는 framework이고, exploit은 취약점 이용 방법, payload는 성공 후 실행 동작, reverse shell은 target이 tester에게 outbound command channel을 여는 형태다.

Metasploit Framework

exploit, payload, auxiliary scanner, post module을 공통 interface로 관리하는 framework다. module target/version과 exploit reliability를 확인하고, production에서는 reverse shell 같은 payload가 EDR·network·service에 미치는 영향을 별도 승인한다.

Payload와 session

exploit은 취약점을 trigger하는 방법이고 payload는 성공 후 실행할 동작이다. session은 framework가 target과 유지하는 control channel이다. 이 셋을 같은 말로 부르면 실패 원인과 risk를 판단하기 어렵다.

Reverse shell

target이 tester의 listener로 outbound connection을 열어 command I/O를 제공하는 shell이다. inbound firewall을 우회할 수 있어 공격에서도 사용된다. 허가된 lab에서만 사용하고 listener address, egress policy, shell privilege, cleanup을 기록한다.

Framework를 쓰지 않는 검증

가능하면 harmless request, version/configuration evidence, controlled file read처럼 최소 impact proof를 먼저 사용한다. remote code execution을 실제 shell로 확장할 필요가 없는 finding도 많다.

37장. 통합 실습과 트러블슈팅: 증상에서 계층별 원인으로

> 학습의 마지막 목표는 용어를 따로 기억하는 것이 아니라, “웹이 안 열린다”, “MAC이 흔들린다”, “IDS alert가 없다” 같은 증상을 packet·table·process·log로 좁히는 것이다.

End-to-End 요청 추적

필기 확인 + 보충 해설

전체 과정을 한 번에 외우지 않고 단계마다 확인 가능한 질문으로 바꾼다

함께 찾을 말: End-to-End Troubleshooting, Dependency Chain

End-to-End 요청 추적란 end-to-end 추적은 사용자 요청을 DNS, L2/L3, transport, TLS, web/application, DB, logging 단계로 나누고 각 단계의 관찰 증거를 연결하는 진단 방법이다.

웹 접속 한 번을 끝까지 추적한다

1. Client가 URL을 parse하고 DNS cache와 resolver를 조회한다.
2. route table로 next hop을 정하고 ARP/ND로 MAC을 찾는다.
3. switch는 VLAN과 MAC table로 frame을 전달한다.
4. router/firewall은 LPM, ACL, NAT, state를 적용한다.
5. server NIC→NAPI→TCP stack→listen socket으로 SYN이 도착한다.
6. TLS handshake가 certificate와 key를 검증한다.
7. web server가 virtual host와 path를 선택하고 WAS/PHP-FPM에 넘긴다.
8. application이 session·authorization을 확인하고 parameterized query로 DB를 조회한다.
9. response가 역경로로 가며 proxy와 browser가 cache·CORS·CSP를 적용한다.
10. 각 장비와 application이 log를 남기고 IDS/Zeek/SIEM이 event를 연결한다.

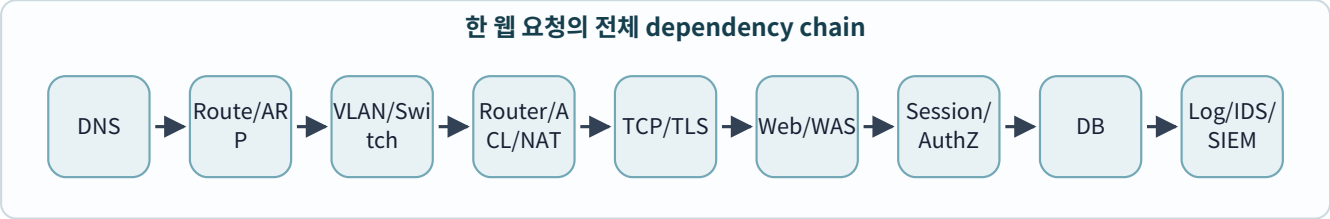


그림 37-1. 앞 단계가 실패하면 뒤 단계의 도구만 보아서는 원인을 찾을 수 없다.

계층별 트러블슈팅

필기 확인 + 보충 해설

설정을 무작정 바꾸기 전에 실패한 첫 단계를 찾는다

함께 찾을 말: Troubleshooting, Layered Diagnosis, Packet Path

계층별 트러블슈팅란 계층별 트러블슈팅은 name resolution, local link, route, transport, security policy, application 순서로 observable evidence를 확인해 장애 범위를 좁히는 방법이다.

“웹이 안 열린다” 확인 순서

단계	질문	도구
이름	DNS가 올바른 address를 주는가	dig, resolvectl
경로	source route와 gateway가 맞는가	ip route get, traceroute
L2	ARP/neighbor와 VLAN이 맞는가	ip neigh, show mac address-table
Transport	SYN/SYN-ACK/RST 중 무엇이 보이는가	tcpdump, ss
Firewall	어느 rule counter가 match하는가	nft list ruleset, ACL counter
TLS	certificate/hostname/version 오류인가	openssl s_client, browser devtools
HTTP	status와 Host/path가 맞는가	curl -v
Application	worker와 DB가 정상인가	journalctl, app log, DB log

Packet capture의 양쪽 비교

client와 server 양쪽에서 같은 5-tuple을 capture하면 packet이 어느 경계에서 사라지는지 알 수 있다. client SYN이 보이지만 server에 없으면 중간 network, server에 SYN은 있지만 SYN-ACK가 없으면 listen/firewall/kernel, server가 SYN-ACK를 보냈지만 client에 없으면 return path/asymmetric routing을 의심한다.

End-to-end 진단 명령 묶음

```
# Client
ip route get 192.0.2.80
ip neigh show
curl -vk --resolve app.example.com:443:192.0.2.80 https://app.example.com/

# Server
ss -lntp | grep ':443'
sudo tcpdump -ni any 'host 192.0.2.10 and tcp port 443'
journalctl -u nginx -u php-fpm --since '10 min ago'
```

L2·Routing·IDS 장애 대응

필기 확인 + 보충 해설

복구와 재발 방지 configuration을 함께 남긴다

함께 찾을 말: L2 Loop Response, Routing Troubleshooting, IDS Troubleshooting

L2·Routing·IDS 장애 대응란 network/security 장애 대응은 증상을 수치와 packet으로 관찰하고, topology·table·state에 대한 가설을 세운 뒤 최소 변경 실험으로 원인을 확인하는 과정이다.

L2 loop 대응

1. 영향 VLAN과 broadcast rate, MAC flapping port를 확인한다.
2. topology와 최근 cable/change를 확인하고 suspect edge/uplink를 하나씩 차단한다.
3. STP root, port role/state, topology change, BPDU guard event를 확인한다.
4. 원인을 제거한 뒤 차단 port를 단계적으로 복구하고 counter를 관찰한다.
5. root guard, loop guard, BPDU guard, storm-control, EtherChannel consistency를 설계한다.

Routing 장애 대응

1. 목적지 prefix가 RIB/FIB에 있는지와 LPM 결과를 본다.
2. next-hop adjacency/ARP가 해결되는지 본다.
3. protocol neighbor와 update/LSDB/topology table을 확인한다.
4. AD, metric, route filter, summary, redistribution을 대조한다.
5. 양방향 return route와 ACL/NAT를 확인한다.

IDS alert 부재 대응

1. SPAN/TAP에서 실제 packet이 sensor NIC에 도착하는지 tcpdump로 확인한다.
2. capture drop, checksum/offload, VLAN tag, snaplen을 확인한다.
3. engine이 correct interface/runmode와 ruleset으로 동작하는지 확인한다.
4. rule header, flow direction, content buffer, SID enable 여부를 test pcap으로 검증한다.
5. alert output pipeline과 SIEM parser/index를 확인한다.

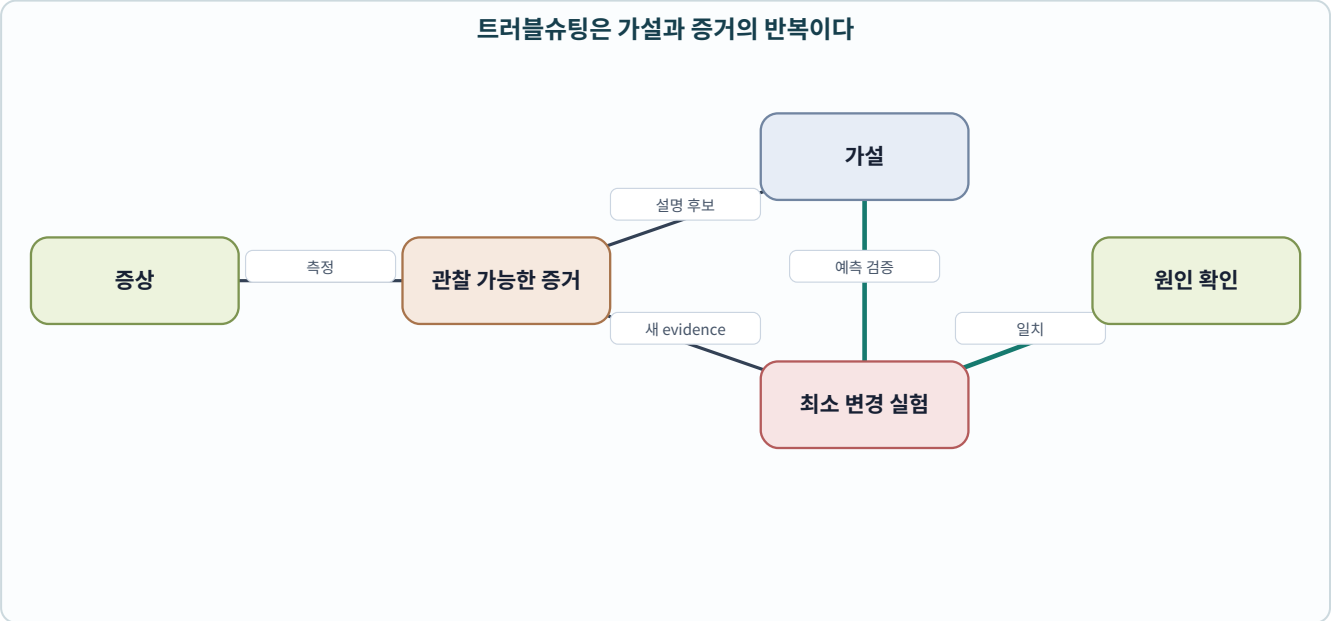


그림 37-2. 한 번에 여러 설정을 바꾸면 어떤 변화가 원인이었는지 알 수 없다.

통합 보안·네트워크 Lab

필기 확인 + 보충 해설

도구 사용 기록이 아니라 packet과 control의 인과관계를 증명한다

함께 찾을 말: Integrated Lab, GNS3 Security Lab, Blue Team Lab, Pentest Lab

통합 보안·네트워크 Lab란 통합 lab은 switching, routing, server, web, monitoring, penetration testing을 하나의 격리 topology에서 연결하고 정상·장애·공격·방어의 evidence를 비교하는 학습 환경이다.

통합 lab 설계

GNS3/VMware host-only network에 router, L2 switch, Linux web/DB, client, Security Onion sensor를 구성한다. 외부 Internet update interface와 lab vulnerable interface를 분리하고, snapshot과 address plan을 문서화한다.

단계별 과제

1. VLAN 10(client), 20(server), 99(management)을 만들고 trunk/802.1Q를 packet capture로 확인한다.
2. STP root를 의도적으로 지정하고 redundant link에서 blocked port를 확인한 뒤 lab에서 loop symptom을 제한적으로 관찰한다.
3. Inter-VLAN routing, DHCP relay, NAT, ACL을 구성하고 packet path를 기록한다.
4. Nginx/PHP-FPM/MariaDB 또는 Flask/WSGI stack을 배포하고 service user·SELinux·firewall을 적용한다.
5. SPAN/TAP로 Suricata/Zeek에 traffic을 보내고 custom safe rule과 protocol log를 검증한다.
6. DVWA/WebGoat에서 SQLi/XSS/CSRF/access control을 재현하고 server-side 방어를 수정한 뒤 재시험한다.
7. rsyslog/Wazuh/SIEM에 device·server log를 모아 incident timeline을 작성한다.

산출물

- 물리·논리 topology와 VLAN/IP/route 표
- configuration version과 변경 근거
- 정상 traffic packet walk와 장애 packet walk
- rule/alert/PCAP/log evidence
- 취약점 finding과 수정 diff·retest 결과
- backup/restore와 incident response runbook

학습 성공 기준

명령이 실행되는 것보다 “이 설정이 어떤 table/state를 만들고, packet이 어느 field를 보고 다음 단계로 가며, 실패하면 어느 evidence가 달라지는가”를 설명할 수 있어야 한다.

APPENDICES

보충 용어·오개념 교정·학습 지도

본문 흐름에서 길게 끼워 넣으면 맥락을 끊지만, 필기에는 등장해 반드시 정확히 짚어야 하는 개념을 모았다. 각 항목도 단어 나열이 아니라 '무엇인가, 왜 필요한가, 실제로 어떻게 동작하는가'를 설명한다.

부록 A. 네트워크 기초 용어를 빈틈없이 연결하기

> 네트워크를 배우다 보면 PAN, LAN, 토폴로지, 허브, 리피터, 유니캐스트 같은 단어가 한꺼번에 쏟아진다. 이 부록은 단어를 외우기보다 “어떤 범위를 가리키는가”, “어느 계층에서 무엇을 복제하거나 변환하는가”를 기준으로 묶는다.

PAN · LAN · CAN · MAN · WAN

필기 확인 + 기초 보강

거리보다 "누가 어디까지 관리하는가"를 함께 본다

함께 찾을 말: Personal Area Network, Local Area Network, Campus Area Network, Metropolitan Area Network, Wide Area Network, 네트워크 범위

PAN · LAN · CAN · MAN · WAN란 네트워크가 덮는 지리적 범위와 관리 범위를 개인 주변, 지역, 캠퍼스, 도시, 광역으로 구분해 부르는 분류다.

범위 이름은 기술 규격이 아니라 관리·지리적 규모를 설명하는 말이다

PAN, LAN, CAN, MAN, WAN은 네트워크가 어느 정도의 거리와 관리 범위를 덮는지 설명한다. 경계가 수학적으로 고정된 것은 아니다. 같은 Ethernet과 IP를 사용해도 한 방 안의 연결이면 PAN/LAN, 여러 건물을 묶으면 CAN, 통신사업자 회선으로 지역·국가를 넘으면 WAN이라고 부를 수 있다.

구분	풀어쓴 이름	전형적인 범위	이해 포인트
PAN	Personal Area Network	한 사람 주변 수 m~수십 m	Bluetooth 이어폰·스마트워치처럼 개인 장치 연결
LAN	Local Area Network	집·사무실·한 건물	조직이 직접 스위치와 배선을 관리하는 경우가 많음
CAN	Campus Area Network	학교·공장·여러 건물	여러 LAN을 캠퍼스 백본으로 묶음
MAN	Metropolitan Area Network	도시권	도시 단위 통신망·사업자 메트로망
WAN	Wide Area Network	지역·국가·대륙	임대회선·MPLS·인터넷 등 원거리 연결

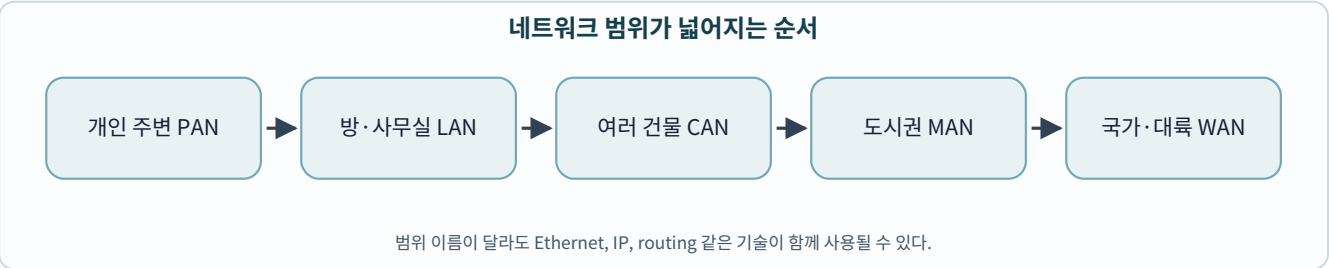


그림 A-1. 범위가 넓어질수록 장비 소유 주체, 회선 제공자, 지연, 장애 구간, 비용을 더 많이 고려한다.

LAN과 Ethernet은 같은 말이 아니다

LAN은 좁은 지역의 네트워크라는 범위·용도 개념이고, **Ethernet**은 그 LAN을 구현하는 대표적인 링크 기술이다. Wi-Fi만으로 만든 LAN도 LAN이며 Ethernet이 아닐 수 있다.

네트워크 토폴로지

필기 확인 + 기초 보강

케이블 모양과 실제 데이터 흐름을 분리해서 본다

함께 찾을 말: Network Topology, 물리 토폴로지, 논리 토폴로지, Bus, Star, Ring, Mesh, Token Passing

네트워크 토폴로지란 네트워크 장치와 링크의 물리적 배치 또는 데이터가 이동하는 논리적 연결 관계를 나타내는 구조다.

토폴로지라는 단어의 뜻

Topology는 “연결 관계의 모양”을 뜻한다. 네트워크에서는 장치와 회선이 어떻게 연결되어 있고, 데이터가 논리적으로 어떤 순서로 이동하는지를 나타낸다. 책상 위 케이블 배치가 별 모양이라고 해서 데이터 흐름도 반드시 별 모양인 것은 아니다.

형태	연결 방식	장점	대표 위험
Bus	하나의 공통 매체에 여러 장치 연결	배선 단순	공통 매체 장애·충돌 영향 큼
Star	각 단말이 중앙 장비에 연결	장애 위치 파악 쉬움	중앙 장비가 단일 장애점이 될 수 있음
Ring	각 장치가 양옆 장치와 고리 형성	순서 있는 접근 가능	고리 단절·우회 설계 필요
Mesh	장치 사이에 여러 경로	높은 복원력	비용·경로 제어 복잡도 증가

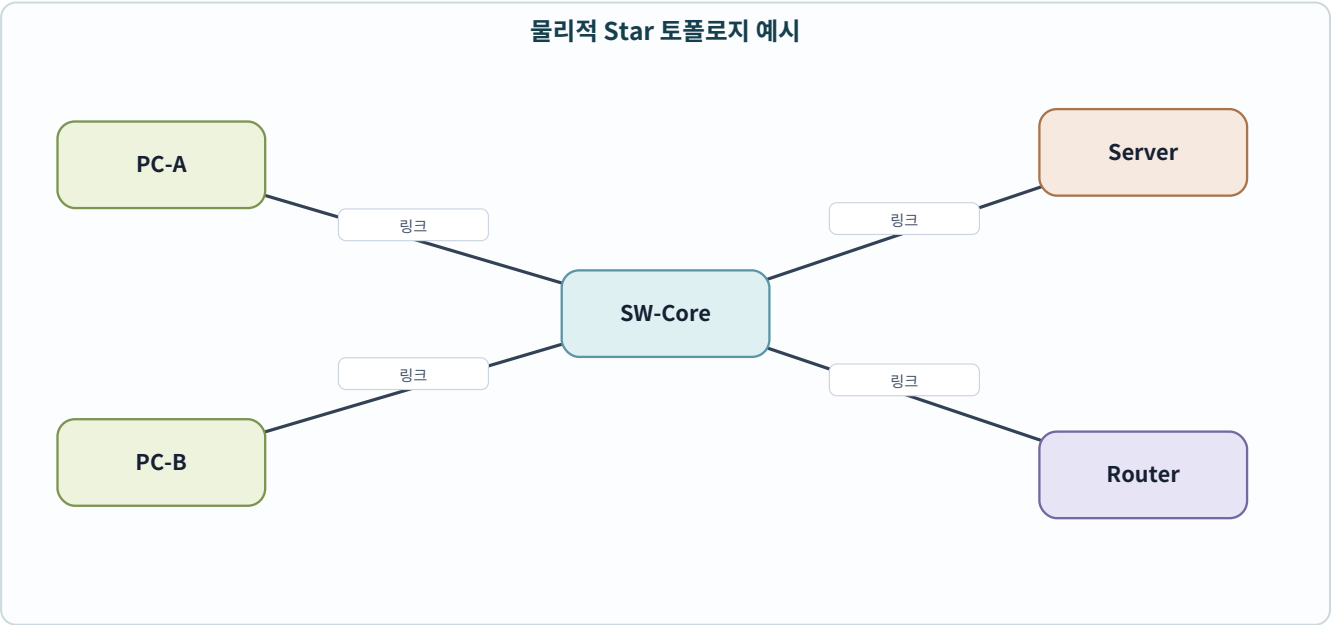


그림 A-2. 모든 장치가 중앙 스위치에 물리적으로 연결된다. 하지만 VLAN·STP·터널에 따라 논리적 데이터 흐름은 다를 수 있다.

물리 토폴로지와 논리 토폴로지

물리 토폴로지는 실제 케이블과 장비의 배치를 나타낸다. 논리 토폴로지는 프레임·토큰·패킷이 어떤 규칙과 경로로 이동하는지를 나타낸다. 예를 들어 과거 Token Ring은 케이블이 중앙 장비를 향하는 별 모양이어도 논리적으로는 토큰이 고리를 돌았다.

토큰은 발언권이다

공유 매체에서 여러 장치가 동시에 말하면 충돌이 생긴다. Token Passing 방식은 “지금 송신해도 되는 권한”을 나타내는 특별한 제어 정보를 순서대로 넘긴다. 토큰을 가진 장치만 전송하므로 충돌을 피할 수 있지만, 토큰 관리와 지연이라는 비용이 있다.

리피터·허브·브리지·스위치·라우터·게이트웨이

필기 확인 + 장비 보강

어느 계층의 어떤 값을 읽는지로 구분한다

함께 찾을 말: Repeater, Hub, Dummy Hub, Bridge, Switch, Router, Gateway, L3 Switch

리피터·허브·브리지·스위치·라우터·게이트웨이란 네트워크 데이터를 물리 신호, MAC 프레임, IP 패킷 또는 서비스 경계 수준에서 각각 재생·복제·선택 전달·경로 선택·변환하는 장비와 역할이다.

리피터는 신호를 되살린다

케이블을 지나며 신호가 약해지고 파형이 흐려지면 먼 거리에서 비트를 구분하기 어렵다. 리피터는 들어온 물리 신호를 재생·정형해 다음 구간으로 내보낸다. MAC 주소나 IP 주소를 읽어 경로를 선택하지 않는다. 따라서 L1 장비다.

허브는 여러 포트가 달린 리피터다

허브는 한 포트에서 받은 비트 신호를 나머지 모든 포트로 반복한다. 어느 목적지인지 판단하지 않으므로 모든 단말이 같은 트래픽을 보고 같은 충돌 도메인을 공유한다. “더미 허브” 또는 “멍청한 허브”라는 표현은 이 무차별 반복 특성을 강조한다.

브리지와 스위치는 프레임을 구분한다

브리지와 스위치는 L2 프레임의 출발지 MAC 주소를 학습하고 목적지 MAC 주소를 조회한다. 그래서 필요한 포트로만 프레임을 전달할 수 있고 포트별 충돌 도메인을 분리한다. 스위치는 많은 포트와 하드웨어 처리 능력을 가진 멀티포트 브리지로 이해할 수 있다.

라우터와 게이트웨이는 “장비 이름”과 “역할”이 다르다

라우터는 IP 네트워크 사이에서 라우팅 테이블을 조회해 패킷을 전달하는 장비·기능이다. 게이트웨이는 한 체계에서 다른 체계로 나가는 관문 역할을 뜻한다. 일반 PC의 default gateway는 보통 같은 LAN에 있는 라우터 인터페이스지만, 프로토콜 변환 게이트웨이처럼 라우터가 아닌 장치도 게이트웨이가 될 수 있다.

대상	주로 보는 정보	전달 방식	도메인 영향
Repeater	전기·광 신호	신호 재생	도메인 분리 안 함
Hub	비트	모든 다른 포트로 반복	하나의 collision domain
Bridge/Switch	MAC 주소	학습한 포트 또는 flooding	포트별 collision domain, VLAN별 broadcast domain

대상	주로 보는 정보	전달 방식	도메인 영향
Router/L3 Switch	IP prefix	라우팅 테이블로 다음 홉 선택	broadcast domain 분리
Gateway	구현에 따라 다름	다른 네트워크·프로토콜·서비스로 중계	역할 개념

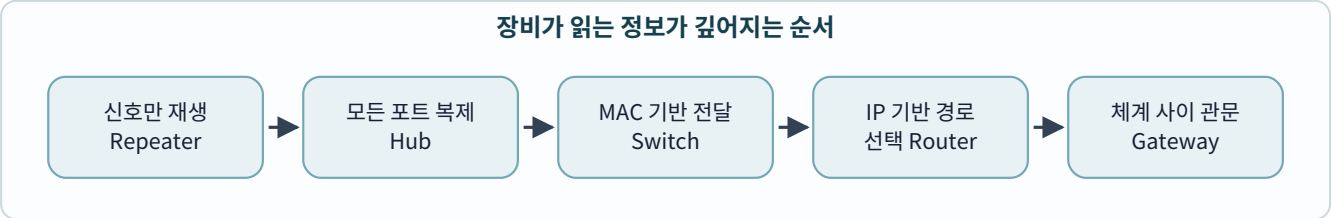


그림 A-3. 오른쪽으로 갈수록 더 높은 계층의 정보와 정책을 사용하지만 실제 장비는 여러 기능을 함께 가질 수 있다.

유니캐스트·브로드캐스트·멀티캐스트·애니캐스트

필기 확인 + IP 보강

"누구에게" 보내는가와 "어디서 복제되는가"를 구분한다

함께 찾을 말: Unicast, Broadcast, Multicast, Anycast, 224.0.0.0/24

유니캐스트·브로드캐스트·멀티캐스트·애니캐스트란 송신 데이터가 한 수신자, 같은 영역의 모두, 특정 가입 그룹, 동일 주소를 제공하는 여러 지점 중 하나를 대상으로 전달되는 통신 방식이다.

유니캐스트

Unicast는 한 송신자가 한 수신자를 대상으로 보내는 방식이다. 웹 서버 한 대에 TCP 연결을 만드는 것이 대표적이다. 다만 스위치가 목적지 MAC 위치를 모르면 유니캐스트 프레임도 일시적으로 flooding할 수 있다.

브로드캐스트

Broadcast는 같은 브로드캐스트 도메인의 모두를 대상으로 한다. Ethernet 브로드캐스트 MAC은 ff:ff:ff:ff:ff:ff이다. ARP Request와 초기 DHCP Discover처럼 아직 상대의 구체 주소를 모를 때 유용하지만, 모든 장치가 받아 처리해야 하므로 범위를 제한해야 한다. 라우터는 기본적으로 L2 브로드캐스트를 다른 네트워크로 전달하지 않는다.

멀티캐스트

Multicast는 특정 그룹에 가입한 여러 수신자에게 한 번의 논리적 송신으로 데이터를 전달한다. 송신자가 수신자 수만큼 복사하지 않아도 네트워크 장비가 필요한 지점에서 복제할 수 있다. IPv4 224.0.0.0/24는 링크 로컬 제어용 멀티캐스트 범위이며 라우팅 프로토콜 등에서 사용한다. 모든 224.0.0.X가 같은 응용이라는 뜻은 아니므로 개별 주소의 표준 용도를 확인해야 한다.

애니캐스트

Anycast는 여러 서버나 라우터가 같은 목적지 주소를 광고하고, 라우팅 관점에서 가장 가까운 또는 가장 선호되는 한 지점으로 트래픽이 가게 하는 방식이다. DNS 서비스와 CDN에서 지연·가용성 개선에 쓰인다. “항상 지리적으로 가장 가까운 서버”가 아니라 라우팅 정책과 경로가 선택한 지점이라는 점이 중요하다.

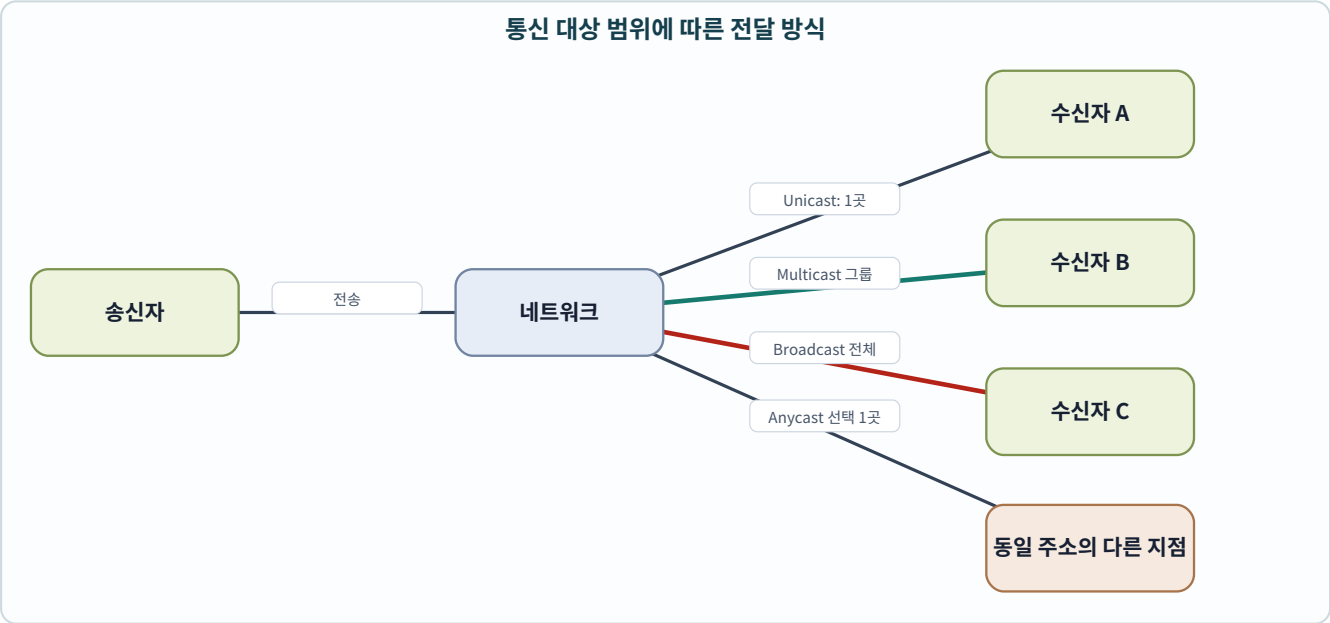


그림 A-4. 송신자의 주소 지정 의도와 네트워크의 복제·경로 선택 방식을 구분한다.

Flooding은 대상 유형이 아니라 스위치의 전달 행동이다

Broadcast는 “누구에게 보내는가”를 뜻하고, flooding은 스위치가 “여러 포트로 복제해 내보내는가”를 뜻한다. Unknown unicast도 flooding될 수 있다.

CSMA/CD와 CSMA/CA

필기 확인 + IEEE 802.3/802.11 보강

공유 회선에서 “언제 말할지” 정하는 규칙

함께 찾을 말: Carrier Sense Multiple Access, Collision Detection, Collision Avoidance, CSMA/CD, CSMA/CA

CSMA/CD와 CSMA/CA란 여러 장치가 공유 매체를 사용할 때 먼저 매체 상태를 확인하고, 유선 반이중 Ethernet은 충돌을 감지해 재시도하며, 무선 LAN은 충돌 가능성을 줄이는 절차를 사용하는 접근 제어 방식이다.

CSMA의 공통 생각

CSMA(Carrier Sense Multiple Access)는 여러 장치가 하나의 매체를 공유할 때 송신 전에 다른 장치가 사용 중인지 감지하는 방식이다. Carrier Sense는 “신호가 들리는가”, Multiple Access는 “여러 장치가 같은 매체에 접근한다”는 뜻이다.

CSMA/CD: 충돌을 발견하고 중단한다

과거 반이중 공유 Ethernet에서는 두 장치가 거의 동시에 송신하면 충돌할 수 있었다. 송신 중 충돌을 감지(Collision Detection)하면 전송을 중단하고 jam signal을 보낸 뒤 임의 시간만큼 기다렸다 재시도한다. 현대의 스위치 기반 전이중 Ethernet에서는 각 링크가 점대점이므로 정상적으로 충돌이 발생하지 않아 CSMA/CD가 사실상 사용되지 않는다.

CSMA/CA: 무선에서는 충돌을 피하려 한다

무선 장치는 송신하면서 동시에 미약한 다른 신호의 충돌을 정확히 감지하기 어렵다. 그래서 802.11은 임의 backoff, ACK, 필요 시 RTS/CTS 같은 절차로 충돌 가능성을 줄이는 Collision Avoidance 방식을 쓴다. 충돌을 완전히 없애는 것이 아니라 발생 가능성과 비용을 낮춘다.

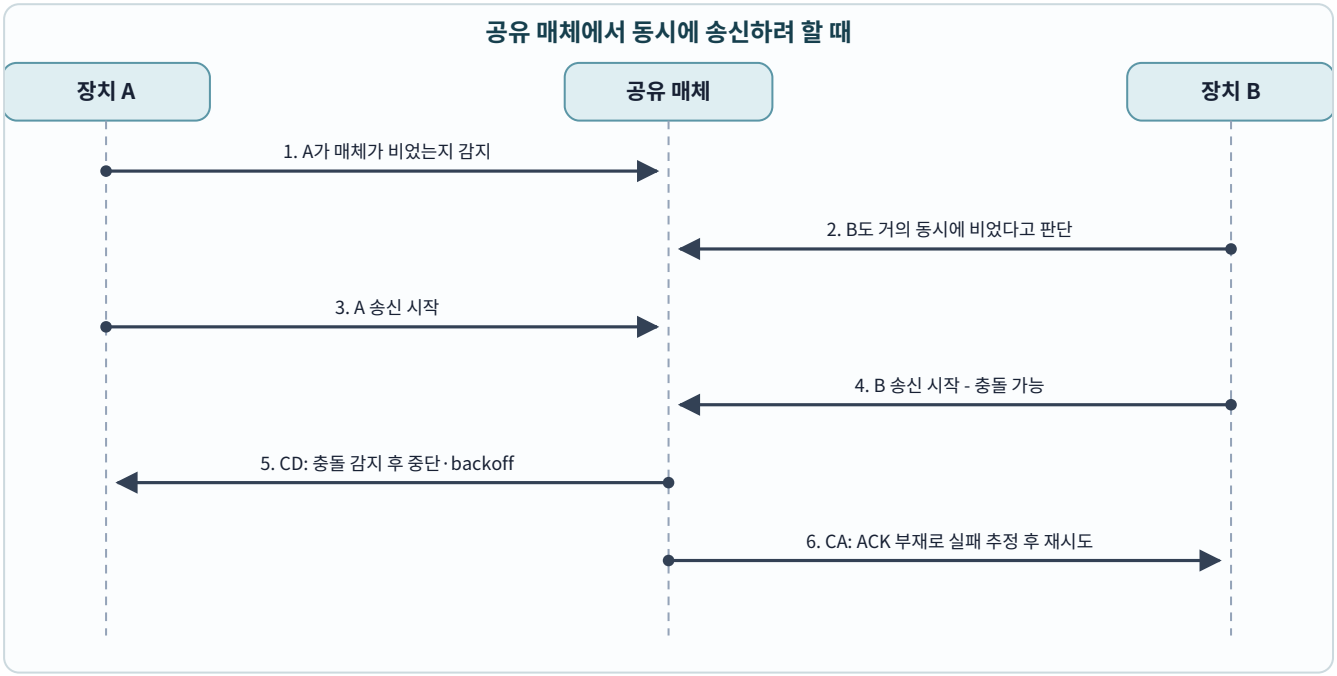


그림 A-5. CD는 송신 중 충돌 감지에, CA는 사전 대기와 확인 응답으로 충돌 회피에 초점이 있다.

Loopback과 127.0.0.1

필기 확인 + IP 보강

자기 자신을 향하는 안정적인 논리 경로

함께 찾을 말: Loopback Interface, 127.0.0.1, localhost

Loopback과 127.0.0.1란 외부 물리 링크를 사용하지 않고 같은 장치의 네트워크 스택으로 되돌아오는 논리 인터페이스와 주소 체계다.

Loopback은 밖으로 나가지 않는 자기 자신과의 통신 경로다

Loopback은 물리 케이블이나 외부 네트워크를 거치지 않고 같은 시스템 내부로 되돌아오는 논리 인터페이스와 주소를 뜻한다. IPv4의 127.0.0.0/8은 loopback 범위이며 일반적으로 127.0.0.1을 localhost로 사용한다.

무엇을 검증할 수 있고 무엇을 검증할 수 없는가

`ping 127.0.0.1`이 성공하면 IP 스택의 기본 동작을 확인할 수 있지만 물리 NIC, 케이블, 스위치, 기본 게이트웨이가 정상이라는 뜻은 아니다. 웹 서버가 127.0.0.1에만 bind되면 같은 호스트에서는 접속되지만 외부에서는 접속할 수 없다.

라우터의 loopback interface

라우터의 loopback interface는 물리 포트와 무관한 논리 인터페이스다. 장비 자체가 살아 있는 한 안정적으로 up 상태를 유지하기 쉬워 router ID, 관리 주소, 라우팅 테스트 기준점으로 사용한다. 단, 그 주소로 가는 라우팅이 있어야 다른 장비에서 도달할 수 있다.

Loopback 확인과 생성

```
ip addr show lo
ping -c 2 127.0.0.1
ss -lntp

# Cisco 계열 예시
interface loopback0
 ip address 10.255.0.1 255.255.255.255
```

부록 B. Linux·Python·자동화에서 자주 만나는 보충 용어

명령어 이름만 외우면 옵션이 조금만 달라져도 막힌다. 이 부록은 shell, 편집기, 링크, 원격 자동화, 객체지향 용어를 “운영체제에서 어떤 자원을 다루는가” 기준으로 설명한다.

bashrc·zshrc·숨김 파일·명령 history

필기 확인 + Shell 보강

편의 설정이 언제 읽히고 어디에 남는가

함께 찾을 말: .bashrc, .zshrc, rc file, hidden file, shell history

> **bashrc·zshrc·숨김 파일·명령 history란** shell이 시작될 때 실행할 사용자 설정과 과거 명령을 저장하는 파일·기능이며, 점으로 시작하는 파일은 단지 기본 목록에서 숨기는 이름 관례다.

rc 파일은 프로그램 시작 시 읽는 설정 명령 모음이다

rc는 역사적으로 run commands라는 뜻으로 쓰여 왔다. Bash의 `~/.bashrc`, Zsh의 `~/.zshrc`에는 alias, 환경 변수, prompt, 함수, 자동완성 설정을 적는다. 어느 파일이 읽히는지는 login shell인지 interactive shell인지에 따라 달라질 수 있으므로 “항상 bashrc 한 곳만 수정”이라는 규칙은 없다.

숨김 파일의 점은 접근 통제가 아니다

Unix 계열에서 이름이 점(.)으로 시작하면 일반 `ls` 출력에서 생략되는 관례가 있다. `ls -a`로 보인다. 숨김 표시는 보안 권한이 아니며 읽기 권한이 있으면 누구든 접근할 수 있다.

history는 편리하지만 민감정보가 남을 수 있다

shell은 입력한 명령을 메모리와 history 파일에 기록할 수 있다. 명령행에 비밀번호·토큰을 직접 넣으면 history, 프로세스 목록, 감사 로그에 남을 수 있다. 가능하면 환경 파일의 권한을 제한하고 stdin, 전용 secret 관리, prompt 입력을 사용한다.

Shell 초기화와 history 확인

```
# 현재 쉘과 설정 파일 확인
echo "$SHELL"
shopt -q login_shell && echo login-shell
history | tail

# 숨김 파일 포함 보기
ls -la ~

# alias 예시
alias ll='ls -alF'
```

cat·head·tail·grep·sed·tee·wget·curl·파이프

필기 확인 + Linux 보강

명령어가 아니라 데이터 흐름을 본다

함께 찾을 말: cat, head, tail, grep, sed, tee, wget, curl, pipe, &&, |

cat·head·tail·grep·sed·tee·wget·curl·파이프란 파일과 표준 입출력 스트림을 읽고, 걸러내고, 변환하고, 복제하고, URL 자원을 전송하는 Unix 도구이며 파이프는 한 프로그램의 출력을 다음 프로그램의 입력으로 연결한다.

파일 내용을 보는 명령은 목적이 다르다

cat

작은 파일을 이어서 표준 출력으로 보낸다. 큰 파일 탐색에는 불편하다.

head/tail

파일의 앞부분·뒷부분만 본다. `tail -f`는 로그에 새 줄이 추가되는 것을 따라간다.

less

큰 파일을 페이지 단위로 탐색하고 검색한다.

grep

패턴에 맞는 줄을 찾는다.

sed

입력 스트림의 텍스트를 규칙에 따라 치환·선택·변환한다.

tee

표준 입력을 화면과 파일에 동시에 복제한다.

&&와 |는 전혀 다른 연결이다

`cmd1 && cmd2`는 첫 명령이 성공(exit status 0)했을 때만 두 번째 명령을 실행하는 제어 연산자다. `cmd1 | cmd2`는 첫 명령의 표준 출력을 두 번째 명령의 표준 입력으로 연결하는 pipeline이다. “앞 명령 결과를 다음 명령에 넘긴다”는 설명은 파이프에만 정확하다.

wget과 curl

둘 다 URL로 데이터를 주고받을 수 있다. wget은 파일 재귀 다운로드와 미러링에 강하고, curl은 HTTP 메서드·헤더·본문·인증 등 요청을 세밀하게 만들고 응답을 확인하는 데 강하다. 실제 서버 점검에서는 curl로 상태 코드와 헤더를 확인하는 경우가 많다.

Shell 연결과 입출력 도구

```
# 파이프: 로그에서 ERROR 줄 수 세기
grep 'ERROR' /var/log/app.log | wc -l

# &&: 패키지 목록 갱신 성공 시 설치
sudo apt update && sudo apt install curl

# tee: 화면에 보면서 root 파일에 쓰기
echo 'example' | sudo tee /etc/example.conf

# HTTP 헤더만 확인
curl -I https://example.com

# 파일 다운로드
wget https://example.com/file.iso
```

vi·Vim·nano와 Ctrl+C

필기 확인 + Unix 보강

편집 모드와 프로세스 신호를 구분한다

함께 찾을 말: vi, vim, nano, SIGINT, Ctrl+C

vi·Vim·nano와 Ctrl+C란 vi/Vim과 nano는 터미널 편집기이며, Ctrl+C는 터미널의 foreground 작업에 인터럽트 신호를 보내 중단을 요청하는 입력이다.

vi와 Vim, nano

vi는 Unix의 전통적인 모드 기반 편집기이며, Vim은 vi 호환 기능을 확장한 편집기다. Normal mode에서 이동·삭제·복사를 하고 Insert mode에서 문자를 입력한다. nano는 화면 아래에 단축키가 표시되는 단순한 편집기로 입문자에게 쉽다. 서버마다 설치 여부가 다르므로 최소한 vi의 저장·종료는 알아두는 편이 좋다.

도구	입력 방식	최소 동작
vi/Vim	Normal/Insert/Command 모드	`i` 입력, `Esc`, `:wq` 저장 종료, `:q!` 강제 종료
nano	즉시 입력	`Ctrl+O` 저장, `Ctrl+X` 종료

Ctrl+C는 “실행 취소 키”라기보다 SIGINT 요청이다

터미널에서 Ctrl+C를 누르면 terminal driver가 foreground process group에 SIGINT 신호를 보낸다. 프로세스는 기본 동작으로 종료되거나, 신호 처리기를 두어 정리 후 종료하거나, 드물게 무시할 수 있다. 따라서 모든 작업이 완벽히 rollback된다는 뜻은 아니다. 파일 쓰기나 DB transaction은 프로그램이 별도로 안전하게 처리해야 한다.

편집기와 중단 신호

```
# vi/Vim 최소 흐름
vi /etc/example.conf
# i -> 편집 -> Esc -> :wq

# nano
nano /etc/example.conf

# 실행 중인 foreground 명령에 SIGINT
# Ctrl+C
```

Hard Link와 Symbolic Link

필기 확인 + filesystem 보강

파일 이름과 실제 데이터의 연결 방식

함께 찾을 말: hard link, symbolic link, symlink, 파일 링크, inode link count

Hard Link와 Symbolic Link란 Hard link는 같은 inode에 연결된 추가 파일 이름이고, symbolic link는 다른 경로를 가리키는 별도 링크 파일이다.

Hard Link는 같은 inode에 이름을 하나 더 붙인다

파일의 실제 메타데이터와 데이터 블록은 inode로 관리되고, directory entry가 파일 이름을 inode 번호에 연결한다. hard link는 같은 inode를 가리키는 다른 이름이다. 한 이름을 삭제해도 link count가 남아 있으면 데이터는 유지된다. 일반적으로 filesystem을 넘을 수 없고 directory hard link는 제한된다.

Symbolic Link는 경로 문자열을 담은 별도 파일이다

symbolic link 또는 symlink는 대상 경로를 내용으로 가진 별도 inode다. 다른 filesystem이나 directory를 가리킬 수 있지만 대상이 이동·삭제되면 깨진 링크가 된다. 권한 점검과 배포 시 링크가 의도하지 않은 위치를 가리키는지 확인해야 한다.

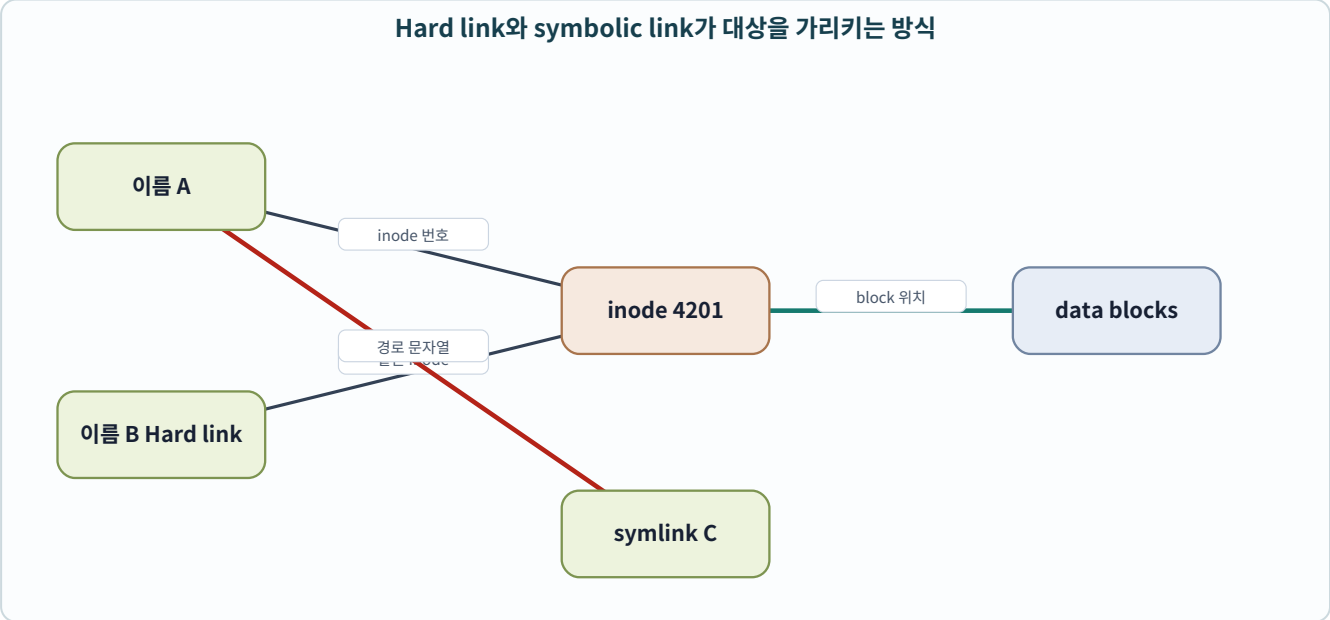


그림 B-1. Hard link는 inode를 공유하고, symbolic link는 다른 이름으로 가는 경로를 저장한다.

링크와 inode 확인

```
echo data > original.txt
ln original.txt hard.txt
ln -s original.txt symbolic.txt
ls -li original.txt hard.txt symbolic.txt
stat original.txt
```

클래스·객체·인스턴스·getter/setter

필기 확인 + Python 보강

객체지향을 문법이 아니라 책임과 경계로 이해한다

함께 찾을 말: class, object, instance, getter, setter, encapsulation, inheritance, composition

클래스·객체·인스턴스·getter/setter란 클래스는 객체가 가질 상태와 동작을 정의하는 타입 설계이고, 인스턴스는 그 클래스에서 만들어진 개별 객체이며, getter/setter는 내부 값을 읽고 검증하며 변경하는 접근 인터페이스다.

클래스는 설계도, 인스턴스는 그 설계로 만들어진 개별 객체다

Class는 공통 상태(attribute)와 동작(method)을 묶어 새로운 type을 정의한다. Object는 프로그램에서 값과 동작을 가진 대상을 넓게 부르는 말이며, 특정 class를 호출해 생성된 개별 object를 instance라고 한다. Python에서는 class 자체도 object다.

캡슐화와 getter/setter

캡슐화는 객체 내부 표현을 외부 코드가 제멋대로 깨뜨리지 않도록 공개 인터페이스를 정하는 설계 원칙이다. getter는 값을 읽는 메서드, setter는 값을 검증해 바꾸는 메서드다. Python에서는 무조건 get_x/set_x를 만들기보다 일반 attribute로 시작하고, 검증이 필요해질 때 `@property`로 인터페이스를 유지하는 방식이 흔하다.

상속보다 구성(composition)이 나을 때

상속은 “A는 B의 한 종류다(is-a)” 관계를 나타낸다. 단순 코드 재사용 때문에 상속 계층을 깊게 만들면 결합도가 높아진다. “A가 B를 가지고 사용한다(has-a)” 관계라면 객체를 attribute로 포함하는 composition이 변경과 테스트에 유리할 수 있다.

Class, instance, property 예시

```
class Port:
    def __init__(self, name: str, vlan: int = 1):
        self.name = name
        self.vlan = vlan

    @property
    def vlan(self) -> int:
        return self._vlan

    @vlan.setter
    def vlan(self, value: int) -> None:
        if not 1 <= value <= 4094:
            raise ValueError("VLAN ID must be 1..4094")
        self._vlan = value

p = Port("Gi0/1", 10)
```

Paramiko·Netmiko·Console·VTY

필기 확인 + 자동화 보강

SSH 연결 계층과 장비 CLI 편의 계층을 분리한다

함께 찾을 말: Paramiko, Netmiko, Console Line, Line VTY, Virtual Teletype, Telnet, SSH automation

Paramiko·Netmiko·Console·VTY란 Paramiko는 Python SSH 구현 라이브러리, Netmiko는 네트워크 장비 CLI 처리를 단순화한 자동화 라이브러리이며, console과 VTY는 각각 로컬·원격 터미널 접속 line이다.

Paramiko는 Python에서 SSH 프로토콜을 다루는 라이브러리다

Paramiko는 SSH 연결, 인증, 채널, SFTP를 Python 코드에서 제어한다. 원격 shell 명령을 실행하고 결과를 받을 수 있지만 vendor별 CLI prompt와 configuration mode를 자동으로 이해해 주는 것은 아니다.

Netmiko는 네트워크 장비 CLI 자동화에 맞춘 상위 도구다

Netmiko는 Paramiko 등을 바탕으로 Cisco, Juniper 등 여러 장비의 prompt, enable mode, command timing, paging 처리를 추상화한다. 장비 설정 자동화에서 편리하지만, 명령 실행 결과가 원하는 상태인지 검증하고 실패 시 rollback·로그를 남기는 설계는 사용자가 해야 한다.

Console, VTY, Telnet, SSH의 관계

Console line은 장비의 로컬 물리/가상 콘솔 접속 경로다. VTY(Virtual Teletype) line은 Telnet/SSH 같은 원격 터미널 세션을 수용하는 논리 line이다. Telnet은 평문이므로 관리망에서도 가급적 SSH를 사용하고, 인증·접근 ACL·로그를 함께 구성한다.

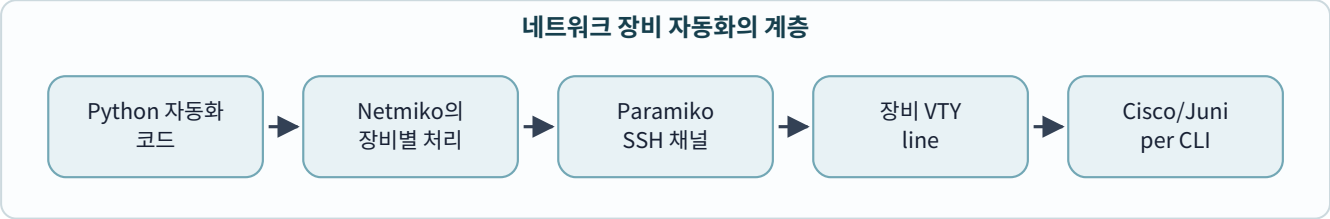


그림 B-2. Netmiko는 장비 CLI 편의 계층이고 Paramiko는 SSH 프로토콜 계층에 가깝다.

허가된 실습 장비에서의 Netmiko 구조 예시

```
from netmiko import ConnectHandler

router = {
    "device_type": "cisco_ios",
    "host": "192.0.2.10",
    "username": "labadmin",
    "password": "<LAB_PASSWORD>",
}

with ConnectHandler(**router) as conn:
    output = conn.send_command("show ip interface brief")
    print(output)
```

자동화는 “명령을 빨리 치는 것”만이 아니다

사전 상태 수집, 변경 계획, idempotency, 예외 처리, 결과 검증, 설정 저장, 백업, 감사 로그가 있어야 운영 자동화가 된다. 비밀번호를 소스 코드에 하드코딩하지 않는다.

Monitorix와 RRD

필기 확인 + 모니터링 보강

실시간 패킷 분석이 아니라 장기 성능 경향을 보는 도구

함께 찾을 말: Monitorix, RRD, RRDtool, MRTG, Cacti

Monitorix와 RRD란 Monitorix는 시스템 상태를 주기적으로 수집해 웹 그래프로 보여주는 경량 모니터링 도구이고, RRD는 고정 크기에서 오래된 시계열 데이터를 단계적으로 집계하는 저장 구조다.

Monitorix는 가벼운 시스템 상태 시각화 도구다

Monitorix는 Linux/Unix 계열 시스템의 CPU, 메모리, filesystem, 네트워크, 서비스 통계를 주기적으로 수집해 웹 그래프로 보여주는 경량 모니터링 도구다. 초 단위 실시간 packet analyzer가 아니라 시간에 따른 경향을 확인하는 도구다.

RRD 구조와 잘 맞는 이유

RRD(Round Robin Database)는 고정 크기의 원형 구조에 시계열 값을 저장하고, 오래된 데이터를 더 낮은 해상도로 집계한다. 저장 공간이 끝없이 커지지 않는 대신 과거 원시값을 영구 보존하는 로그 DB와 목적이 다르다. Monitorix나 MRTG, Cacti 계열 도구는 이런 특성을 이용해 장기간 그래프를 만든다.

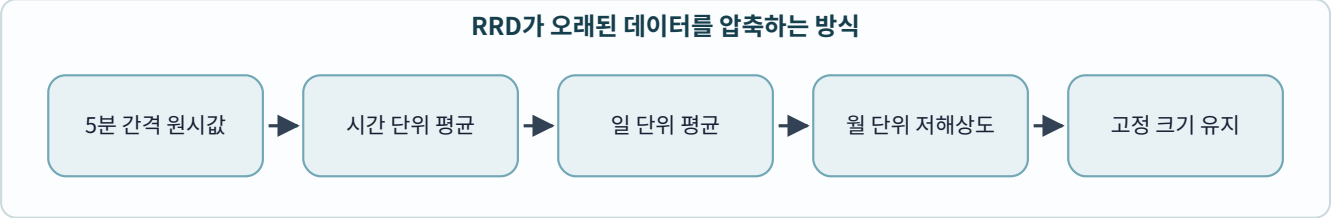


그림 B-3. 최근 데이터는 세밀하게, 오래된 데이터는 집계해 제한된 저장 공간으로 장기 추세를 보존한다.

Mainframe·Workstation·VDI

필기 확인 + 인프라 보강

컴퓨터의 크기보다 어떤 업무와 운영 모델을 위해 설계했는가

함께 찾을 말: Mainframe, Workstation, Virtual Desktop Infrastructure, VDI

Mainframe·Workstation·VDI란 대규모 중앙 업무 처리 시스템, 개인 전문 고성능 컴퓨터, 중앙 VM으로 사용자 desktop을 제공하는 인프라를 각각 가리킨다.

Mainframe

Mainframe은 대규모 조직의 핵심 거래를 높은 신뢰성·처리량·가용성으로 처리하도록 설계된 중앙 시스템 계열이다. 단순히 “아주 큰 PC”가 아니라 I/O 처리, 파티셔닝, 장애 격리, 장기 호환성을 중시한다.

Workstation

Workstation은 CAD, 과학 계산, 3D, 개발처럼 한 사용자의 전문 작업에 높은 CPU/GPU·메모리·신뢰성을 제공하는 컴퓨터다. 오늘날 고성능 PC와 경계가 겹치지만 인증된 부품, ECC 메모리, 전문 GPU 드라이버를 강조할 수 있다.

VDI

VDI(Virtual Desktop Infrastructure)는 사용자 desktop OS를 데이터센터의 VM으로 실행하고 화면·키보드·마우스 입력을 원격 전송한다. 단말에 데이터가 덜 남고 중앙 관리가 쉬운 장점이 있지만, 접속망 지연·스토리지 IOPS·로그인 폭주·라이선스가 중요한 설계 요소다.

개념	주 사용 관점	핵심 설계 요소
Mainframe	대규모 중앙 거래 처리	가용성·I/O·호환성·파티션
Workstation	개인 전문 고성능 작업	CPU/GPU·메모리·인증 드라이버
VDI	중앙에서 desktop 제공	세션 밀도·스토리지·지연·접근 통제

부록 C. 표준 문서와 프로토콜 표기 읽는 법

> IEEE, ISO, IETF, RFC, TLV 같은 말은 프로토콜 자체와 표준을 만든 조직·문서 형식을 섞어 쓰기 쉽다. 어떤 문서를 어디서 확인해야 하는지와 필드 표기법을 정리한다.

ISO · IEEE · IETF · RFC

필기 확인 + 공식 문서 보강

조직과 문서와 프로토콜을 구분한다

함께 찾을 말: ISO, IEEE, IETF, RFC, Request for Comments, Internet Standard

ISO · IEEE · IETF · RFC란 국제 표준 기구, 전기·전자·LAN 표준 전문 단체, 인터넷 표준화 공동체, 그리고 인터넷 기술 사양을 기록하는 문서 시리즈를 각각 가리킨다.

ISO

ISO(International Organization for Standardization)는 여러 산업 분야의 국제 표준을 만드는 독립 국제기구다. OSI Reference Model은 ISO 표준 체계에서 네트워크 기능을 7개 계층으로 설명하는 참조 모델이다. ISO가 인터넷의 모든 프로토콜 구현을 직접 정한다는 뜻은 아니다.

IEEE

IEEE(Institute of Electrical and Electronics Engineers)는 전기·전자·컴퓨팅 분야의 전문 단체이며 IEEE 802 계열에서 LAN/MAN 기술을 표준화한다. Ethernet은 주로 IEEE 802.3, 무선 LAN은 802.11, bridge·VLAN은 802.1Q 계열과 연결된다.

IETF와 RFC

IETF는 인터넷 기술 표준화 공동체이며 핵심 결과물을 RFC(Request for Comments) 시리즈로 발행한다. RFC에는 Internet Standard뿐 아니라 Best Current Practice, Informational, Experimental, Historic 문서도 있다. 번호가 있다는 이유만으로 모두 현재 표준은 아니므로 상태, 업데이트·폐기 관계, errata를 확인해야 한다.

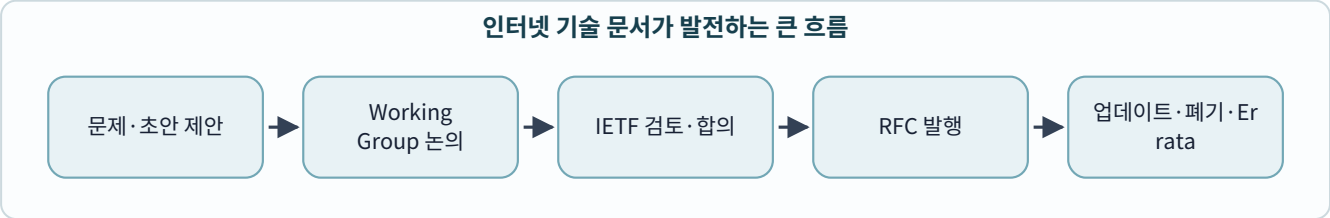


그림 C-1. 실제 절차는 더 복잡하며 모든 RFC가 같은 표준 단계에 있는 것은 아니다.

“RFC에 있다 = 현재 반드시 구현해야 하는 표준”은 아니다

RFC 문서 상단의 category와 status, “Updates”, “Obsoletes” 관계를 확인한다. TCP의 현재 핵심 사양은 RFC 9293처럼 과거 문서를 통합·대체한 문서를 우선 봐야 한다.

TLV

필기 확인 + 프로토콜 보강

모르는 옵션도 길이를 보고 건너뛸 수 있게 하는 포맷

함께 찾을 말: Type-Length-Value, TLV, sub-TLV

> **TLV란** Type으로 값의 종류를 표시하고, Length로 길이를 알리며, Value에 실제 데이터를 넣는 확장 가능한 데이터 인코딩 구조다.

Type

이 값 묶음이 무엇을 뜻하는지 나타내는 식별자다. 예를 들어 “이 TLV는 주소 정보”, “이 TLV는 metric 정보”처럼 해석 방식을 선택하게 한다.

Length

Value 영역의 길이를 알려준다. 수신자는 다음 필드 경계를 계산하고 자신이 모르는 type도 길이만큼 건너뛸 수 있다. 길이 검증이 부실하면 buffer over-read, parsing 오류, DoS 취약점이 생길 수 있다.

Value

실제 주소, 이름, 옵션, metric 같은 데이터다. Type마다 내부 형식이 다르다. 일부 프로토콜은 TLV 안에 다시 sub-TLV를 넣어 확장한다.

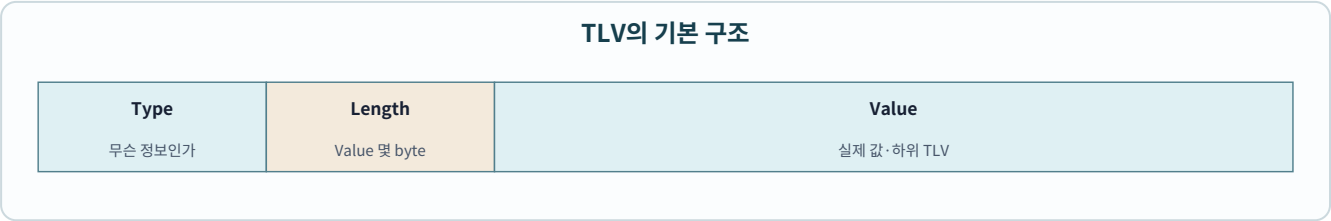


그림 C-2. Length 덕분에 수신자가 모르는 Type도 경계를 잃지 않고 건너뛸 수 있어 확장성이 높다.

어디에서 보는가

LLDP, IS-IS, RADIUS attribute, DHCP option 등 여러 프로토콜이 type-length-value 또는 유사한 구조를 사용한다. 다만 Type와 Length의 비트 수, Length에 header를 포함하는지, byte order는 프로토콜마다 다르므로 해당 표준을 확인해야 한다.

EtherType · IP Protocol Number · Port Number

필기 확인 + IANA/RFC 보강

숫자가 같아 보여도 계층과 의미가 다르다

함께 찾을 말: EtherType, Protocol Number, Next Header, Port Number, Well-known Port

EtherType · IP Protocol Number · Port Number란 각 계층의 header에서 다음에 올 프로토콜이나 호스트 내부 통신 endpoint를 식별하는 서로 다른 번호 공간이다.

EtherType는 Ethernet payload의 상위 프로토콜을 알려준다

Ethernet II frame의 Type 필드는 뒤 payload가 IPv4, ARP, IPv6, 802.1Q tagged frame 중 무엇인지 식별한다. 대표적으로 IPv4는 0x0800, ARP는 0x0806, IPv6는 0x86DD, 802.1Q tag 존재를 나타내는 TPID는 흔히 0x8100이다.

Protocol Number는 IP payload를 알려준다

IPv4 header의 Protocol과 IPv6의 Next Header는 뒤에 TCP, UDP, ICMP 등이 오는지 식별한다. TCP는 6, UDP는 17처럼 번호가 정해져 있다. 이 번호는 TCP/UDP port 번호와 다른 namespace다.

Port Number는 호스트 안의 통신 endpoint를 구분한다

TCP/UDP port는 IP 주소로 도착한 데이터를 어느 socket과 process에 전달할지 구분하는 16bit 번호다. 같은 숫자 53을 TCP와 UDP가 각각 별도 namespace로 사용할 수 있다. “TCP와 UDP는 포트를 공유하지 않는다”는 말은 이 독립성을 뜻한다.

식별자	어디에 있음	무엇을 구분
EtherType	Ethernet header	상위 L3/링크 프로토콜
IP Protocol / Next Header	IP header	TCP·UDP·ICMP 등 다음 header
TCP/UDP Port	Transport header	호스트의 socket·service
VLAN ID	802.1Q tag	L2 broadcast domain

부록 D. 자주 생기는 오개념 30선

> 입문 필기는 빠르게 적기 때문에 “대충 맞지만 범위가 잘못된 문장”이 생기기 쉽다. 아래 표는 틀린 문장을 외우지 않도록 경계를 바로잡는다. 각 교정의 상세 메커니즘은 본문 용어 색인으로 이동해 확인한다.

흔한 오해	정확한 교정
“MAC 주소 표는 ARP 표다”	MAC 주소 표는 VLAN+MAC→스위치 포트이고, ARP/neighbor 표는 IP→MAC이다.
“Gateway는 곧 Router라는 장비 이름이다”	Gateway는 다른 영역으로 나가는 관문 역할이며, default gateway를 라우터 인터페이스가 맡는 경우가 흔하다.
“Port Security는 포트를 암호화한다”	Port Security는 switch port에서 허용할 source MAC과 개수, 위반 동작을 제어한다. 암호화 기능이 아니다.
“802.1Q는 패킷에 VLAN 번호만 붙인다”	Ethernet frame에 4-byte tag를 삽입하며 TPID, PCP, DEI, VID 필드를 사용한다. IP packet 안에 넣는 것이 아니다.
“L2 loop는 같은 frame 하나가 원을 도는 것뿐이다”	broadcast/unknown unicast가 각 switch에서 복제되어 여러 사본이 돌고, MAC table도 계속 뒤집히므로 지수적으로 악화될 수 있다.
“Ethernet에는 TTL이 있으니 loop가 끝난다”	일반 Ethernet frame에는 IP TTL 같은 hop 제한이 없다. STP 등으로 active topology를 loop-free하게 만들어야 한다.
“STP는 여분 케이블을 끈다”	포트를 논리적으로 discarding/blocking해 data frame 전달을 막지만 BPDU를 통해 topology를 감시하고 장애 시 재계산한다.
“Native VLAN은 항상 VLAN 1이다”	많은 장비의 기본값이 VLAN 1일 수 있으나 native VLAN은 trunk에서 untagged frame을 연관시키는 설정이며 변경 가능하다.
“Trunk port는 모든 VLAN을 무조건 통과시킨다”	허용 VLAN 목록, native VLAN, pruning, STP 상태, 보안 정책에 따라 전달 범위가 제한된다.
“ARP는 Router를 찾는 프로토콜이다”	ARP는 같은 링크에서 next-hop IPv4 주소에 대응하는 MAC을 찾는다. next hop이 목적지 host일 수도 gateway일 수도 있다.
“서브넷 마스크가 통신을 허용하거나 차단한다”	마스크는 주소의 prefix/host 경계를 해석해 local/remote 판단과 route matching에 사용한다. 접근 통제는 firewall/ACL 등이 한다.
“NAT는 방화벽이다”	NAT는 주소·port 변환이고 firewall은 정책에 따른 허용/차단이다. stateful NAT가 우연히 inbound를 어렵게 만들 수 있지만 같은 기능은 아니다.
“TCP handshake가 끝나면 데이터는 절대 유실되지 않는다”	TCP는 재전송·순서 제어로 신뢰성 있는 byte stream을 제공하지만 timeout, reset, endpoint 장애, application 오류까지 없애지는 않는다.

흔한 오해	정확한 교정
“UDP는 무조건 빠르고 TCP는 무조건 느리다”	부가 제어 차이는 있으나 실제 지연은 application 설계, 손실, congestion, handshake, kernel 구현에 따라 달라진다.
“Ping이 되면 서비스도 정상이다”	ICMP echo와 TCP/UDP service는 별도 protocol·port·process·firewall 경로를 사용한다.
“127.0.0.1 ping 성공은 NIC와 케이블 정상이다”	Loopback은 host 내부 stack만 확인하며 외부 물리 경로를 검증하지 않는다.
“RAID는 backup이다”	RAID는 disk 장애 가용성과 성능을 위한 배열이며 삭제·암호화·논리 손상·재해를 별도 시점으로 복구하지 못한다.
“chmod 777이면 권한 문제는 해결된다”	모든 사용자에게 쓰기 권한을 열어 무결성·권한 상승 위험을 만든다. owner/group, 최소 권한, SELinux context를 함께 봐야 한다.
“SELinux permissive는 SELinux가 꺼진 상태다”	정책 위반을 기록하지만 차단하지 않는 모드다. disabled와 다르며 원인 분석에 활용한다.
“bind address는 접속 허용 목록이다”	service가 어느 local IP에서 listen할지 정하는 값이다. remote source 허용은 firewall·DB account host·application ACL이 별도로 담당한다.
“Hash는 암호화라서 key로 복호화할 수 있다”	Hash는 임의 길이 입력을 고정 길이 digest로 매핑하는 단방향 함수다. password 저장은 salt와 느린 password hashing/KDF를 사용한다.
“IDS와 IPS는 같은 제품의 이름 차이뿐이다”	IDS는 관찰·경보, IPS는 inline 경로에서 차단 가능성을 가진다. 배치와 failure mode가 다르다.
“SPAN으로 wire의 모든 것을 완벽히 본다”	oversubscription, switch 처리, error frame/FCS 제거, filtering 때문에 누락될 수 있다. TAP과 endpoint capture를 목적에 맞게 병행한다.
“Promiscuous mode면 LAN의 모든 packet을 볼 수 있다”	NIC가 자기 MAC 이외 frame도 OS에 넘기게 하지만 switched network에서 다른 port traffic이 자동으로 도착하는 것은 아니다.
“CVE는 취약점 원인 분류다”	CVE는 공개 취약점 사례 식별자이고 CWE는 원인이 되는 약점 유형 분류다.
“OWASP Top 10은 취약점 전체 목록이다”	웹 application 위험 인식을 위한 상위 category 문서이며 모든 CWE와 공격 유형을 포괄하는 checklist가 아니다.
“Nmap 결과 open은 취약함을 뜻한다”	해당 transport probe에 service가 응답했다는 뜻이다. 취약성은 version, configuration, exposure, authentication을 추가 평가해야 한다.
“룰이 많을수록 IDS가 정확하다”	중복·부적절한 rule은 false positive와 성능 저하를 늘린다. 자산·traffic baseline·tuning이 필요하다.
“로그를 많이 저장하면 관제가 된다”	시간 동기화, field normalization, detection logic, 담당자 triage, response 절차가 있어야 사건으로 해석할 수 있다.
“자동화는 명령을 반복 실행하는 script다”	사전 검증, idempotency, 오류 처리, rollback, 결과 확인, secret 관리, 감사 추적까지 포함해야 운영 자동화다.

부록 E. 75일 필기를 개념 흐름으로 다시 배치한 학습 지도

> 원본 필기는 날짜순으로 진행되었지만 복습은 개념 의존성 순서가 더 효율적이다. 아래 표는 실제 학습 시기를 보존하면서 같은 개념을 한 장에 모은 재배치 지도다. 프로젝트만 진행한 날은 해당 프로젝트가 요구한 기술 묶음에 연결했다.

원본 학습 시기	필기에서 확인된 중심 내용	이 교재에서 먼저 읽을 곳	연결해서 읽을 곳
1~4일차	VM, ISO, Linux shell, DNS/cache, 권한, GNS3, 네트워크 범위	1~5장	부록 A·B
5~13일차	Python 자료형, 참조, 조건·반복, 함수, scope, 파일 I/O	1~4장	2장 메모리, 부록 B 객체지향
14~20일차	OSI, MAC, Ethernet, switch, ARP, IP, routing, TCP/UDP	6~13장	부록 A·C
21~29일차	DHCP, NAT, virtualization, Telnet/SSH, static route, VLAN, STP, VTP, EtherChannel	5장·12~16장·8~9장	부록 A의 장비·통신 방식
30~39일차	서버 hardware, LAMP, web/WAS, DNS, account, storage, RAID/LVM, logging	18~23장	3장 Linux, 24~26장 보안
40~51일차	통합 프로젝트 구축·연동·문서화	37장	18~23장과 31장
52~54일차	Frame Relay, serial, HDLC/PPP, ACL, ARP 변형, proxy/MITM	17장·12장·33장	24장 암호·신뢰 경계
55~61일차	모의해킹 절차·도구, monitoring, packet capture, IDS/IPS, OWASP, SNMP	28~34장	23장 logging, 31장 표준
62~68일차	RIP, XSS/session, Linux RX path, firewall, Security Onion, distribute-list	15장·27~30장·34~35장	16장 고급 routing, 26장 Netfilter
69~75일차	Snort rules, Nmap/OSINT/Nessus, EIGRP, Netfilter/firewalld/UFW, IGP/EGP	29~32장·16장·26장	37장 통합 troubleshooting

복습 권장 순서

1회차: 각 장의 정의 상자와 그림만 본다. **2회차**: 동작 순서를 손으로 다시 그린다. **3회차**: 실습 명령을 실행하고 "어느 계층의 어떤 상태를 확인하는가"를 설명한다. **4회차**: 장애 시나리오를 주고 아래 계층부터 확인한다.

부록 F. 공식 문서 중심 참고자료

> 본문은 필기에서 확인한 학습 범위를 뼈대로 삼고, 표준 동작과 현재 보안 분류는 아래 공식 문서로 교차 확인했다. 제품 명령은 버전·platform에 따라 달라질 수 있으므로 실제 장비의 release 문서를 우선한다.

1. **IEEE 802.1 Working Group - IEEE Std 802.1Q-2022 revision**: https://1.ieee802.org/maintenance/802-1q-2022-rev/ - Bridges, VLANs, bridged networks
2. **IEEE 802.3 Ethernet Working Group**: https://www.ieee802.org/3/ - Ethernet standards and projects
3. **Cisco - Port Security Configuration Guide**: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/12-2_58_se/configuration/guide/2960scg/swtrafc.html - secure MAC and violation behavior; command details vary by platform
4. **Cisco - STP and Layer 2 loop troubleshooting**: https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/10556-16.html - Spanning Tree concepts and troubleshooting
5. **RFC Editor**: https://www.rfc-editor.org/ - Authoritative RFC publication index
6. **RFC 826 - An Ethernet Address Resolution Protocol**: https://www.rfc-editor.org/rfc/rfc826 - ARP
7. **RFC 1918 - Address Allocation for Private Internets**: https://www.rfc-editor.org/rfc/rfc1918 - IPv4 private address space
8. **RFC 2131 - Dynamic Host Configuration Protocol**: https://www.rfc-editor.org/rfc/rfc2131 - DHCP
9. **RFC 2328 - OSPF Version 2**: https://www.rfc-editor.org/rfc/rfc2328 - OSPFv2
10. **RFC 2453 - RIP Version 2**: https://www.rfc-editor.org/rfc/rfc2453 - RIPv2
11. **RFC 4271 - BGP-4**: https://www.rfc-editor.org/rfc/rfc4271 - BGP-4
12. **RFC 8200 - Internet Protocol, Version 6**: https://www.rfc-editor.org/rfc/rfc8200 - IPv6
13. **RFC 8446 - The Transport Layer Security Protocol Version 1.3**: https://www.rfc-editor.org/rfc/rfc8446 - TLS 1.3
14. **RFC 9293 - Transmission Control Protocol**: https://www.rfc-editor.org/rfc/rfc9293 - Current consolidated TCP specification
15. **IANA Protocol Registries**: https://www.iana.org/protocols - Port numbers, protocol numbers, address registries
16. **Linux kernel networking documentation**: https://docs.kernel.org/networking/ - NAPI, sk_buff and network stack documentation
17. **Linux nftables Wiki**: https://wiki.nftables.org/ - nftables concepts and examples
18. **firewalld documentation**: https://firewalld.org/documentation/ - Zones, services, rich rules
19. **Python 3 Data Model**: https://docs.python.org/3/reference/datamodel.html - objects, identity, type, value, mutability
20. **Python Tutorial**: https://docs.python.org/3/tutorial/ - language and standard library foundations
21. **MariaDB Server Documentation**: https://mariadb.com/kb/en/documentation/ - accounts, SQL, server configuration
22. **Apache HTTP Server Documentation**: https://httpd.apache.org/docs/ - virtual hosts, modules, access control
23. **NGINX Documentation**: https://nginx.org/en/docs/ - HTTP server and reverse proxy
24. **Snort Documentation**: https://docs.snort.org/ - rules, DAQ, packet processing
25. **Suricata User Guide**: https://docs.suricata.io/ - IDS/IPS/NSM engine
26. **Zeek Documentation**: https://docs.zeek.org/ - network security monitoring and logs
27. **Security Onion Documentation**: https://docs.securityonion.net/ - integrated NSM platform
28. **OWASP Top 10:2025**: https://owasp.org/Top10/2025/ - web application security awareness categories
29. **OWASP Web Security Testing Guide**: https://owasp.org/www-project-web-security-testing-guide/ - authorized web security test methodology
30. **MITRE ATT&CK Updates**: https://attack.mitre.org/resources/updates/ - current ATT&CK version and change history
31. **NIST SP 800-115**: https://csrc.nist.gov/pubs/sp/800/115/final - technical security testing and assessment guide
32. **MITRE CVE**: https://www.cve.org/ - public vulnerability identifiers

33. **MITRE CWE**: https://cwe.mitre.org/ - software and hardware weakness types
34. **KISA 보호나라 & KrCERT**: https://www.boho.or.kr/ - Korean security advisories and guidance

문서를 확인하는 순서

프로토콜은 RFC/IEEE/IANA → 운영체제는 kernel·project 공식 문서 → 제품 명령은 해당 release configuration guide → 보안 분류는 OWASP/MITRE/NIST/KISA 순으로 원문을 확인한다. 블로그는 이해를 돕는 2차 자료로만 사용하고 표준 판단 근거로 삼지 않는다.

용어 찾아보기

본문은 문자 검색이 가능하다. 아래 목록은 표준 영문명, 약어, 한글 표현을 함께 찾기 위한 색인이다.

A-Z / 0-9

127.0.0.1 → Loopback과 127.0.0.1

16진수 → 비트·바이트·옥텟과 진법

224.0.0.0/24 → 유니캐스트·브로드캐스트·멀티캐스트·애니캐스트

2진수 → 비트·바이트·옥텟과 진법

3-way handshake → TCP 연결 설정과 종료

4-way handshake → TCP 연결 설정과 종료

5-tuple → Port · Socket · 5-tuple

802.1Q → IEEE 802.1Q (dot1q)

802.1w → STP 포트 상태와 RSTP

AAAA → DNS Record · TTL · 정방향/역방향 조회

accept → Port · Socket · 5-tuple

Access-Control-Allow-Origin → Same-Origin Policy · CORS · Origin

Access Control List → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Access port와 Trunk port

Accounting → CIA · 인증·인가·최소 권한

ACK → TCP 연결 설정과 종료

Acknowledgment Number → TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

ACL → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Active Response → EDR · XDR · SOAR · 능동 대응

AD → Static · Dynamic Routing · Metric · AD · ECMP

Address Resolution Protocol → ARP

Adjacency → Routing Table · Longest Prefix Match · Gateway

Adjacent → Routing Table · Longest Prefix Match · Gateway

Administrative Distance → Static · Dynamic Routing · Metric · AD · ECMP

AEAD → 대칭키·비대칭키·Hash · Digital Signature

AES → 대칭키·비대칭키·Hash · Digital Signature

AF_PACKET → BPF · eBPF · AF_PACKET · DPDK

Aircrack-ng → Burp Suite · sqlmap · Hydra · John · Hashcat

Alternate Port → STP 포트 역할

Anaconda → 실행 인자와 Python 가상 환경

ANSI → 문자 인코딩과 ASCII

Ansible → On-premises · Cloud · IaC · Container

Anycast → 유니캐스트·브로드캐스트·멀티캐스트·애니캐스트

Apache → LAMP · Web Server · WAS

append → 파일 입출력·Context Manager · Race Condition

apt → 패키지 관리자·Repository · upstream/downstream

Area → OSPF · Link State · Dijkstra

A Record → DNS Record · TTL · 정방향/역방향 조회

Argon2 → Hash · Salt · Rainbow Table

argparse → 실행 인자와 Python 가상 환경

args → 함수·메서드·인자·1급 객체

argument → 함수·메서드·인자·1급 객체

argv → 실행 인자와 Python 가상 환경

ARP

ARP cache → ARP

ARP Spoofing → Sniffing · Spoofing · MITM

ARP Spoofing 방어 → Dynamic ARP Inspection

ARQ → 오류 제어·ARQ · Hamming Code

ASCII → 문자 인코딩과 ASCII

ASGI → CGI · FastCGI · PHP-FPM · WSGI · Worker

ASN → IGP · EGP · AS

AS_PATH → BGP · Path Vector

Asset → 자산·위협·취약점·위험

ASVS → OWASP · KISA · 주요정보통신기반시설 가이드

Asymmetric Cryptography → 대칭키·비대칭키·Hash · Digital Signature

Attack Surface → 자산·위협·취약점·위험

Authentication → CIA · 인증·인가·최소 권한

Authoritative Server → DNS · Domain · FQDN · Zone

Authorization → CIA · 인증·인가·최소 권한

Autonomous System → IGP · EGP · AS

Autopsy → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

Auto Summary → RIP와 Distance Vector

Availability → CIA · 인증·인가·최소 권한

AVC → SELinux

Bandwidth → 대역폭·지연·Duplex

Bare metal → 가상화·Hypervisor·VM

Barnyard2 → NSM·Security Onion·Sguil/Squert/Barnyard2

bashrc·zshrc·숨김 파일·명령 history

bcrypt → Hash·Salt·Rainbow Table

BDR → OSPF·Link State·Dijkstra

bee-box → Kali Linux·Metasploitable·DVWA·WebGoat·bee-box

BeEF → Burp Suite·sqlmap·Hydra·John·Hashcat

Berkeley Packet Filter → BPF·eBPF·AF_PACKET·DPDK

BGP → BGP·Path Vector

BGP·Path Vector

BID → 루트 브리지와 Bridge ID

binary → 비트·바이트·옥텟과 진법

Binary Log → CMS와 DB 운영

Binary Logging → Logrotate·NTP·Serialization

BIND → BIND·named·DNSSEC·dig

bind → Port·Socket·5-tuple

bind-address → DB 계정·Host·Bind Address

BIND·named·DNSSEC·dig

bind-utils → BIND·named·DNSSEC·dig

BIOS → 서버 부팅·UEFI·MBR·GPT·Partition

bit → 비트·바이트·옥텟과 진법

Blackhole → Hop·Convergence·Routing Loop·Stub Network

Blade Server → Rack Server·Blade Server·BMC/iLO

Blind SQL Injection → SQL Injection·Blind SQL Injection

Block Device → Block Device·Local/Remote Storage·SCSI/NVMe

Block Device·Local/Remote Storage·SCSI/NVMe

Blocking → STP 포트 상태와 RSTP

Blocking Port → STP 포트 역할

Blue Team Lab → 통합 보안·네트워크 Lab

BMA → NBMA 환경과 RIP

BMC → Rack Server·Blade Server·BMC/iLO

BOLA → Broken Access Control·IDOR/BOLA

Bootloader → 서버 부팅·UEFI·MBR·GPT·Partition

Boot Process → 서버 부팅·UEFI·MBR·GPT·Partition

BPDU

BPDU Guard → STP 보호 기능

BPF → BPF·eBPF·AF_PACKET·DPDK

BPFDoor → BPF·eBPF·AF_PACKET·DPDK

BPF·eBPF·AF_PACKET·DPDK

break → 조건문·반복문·종료 조건

Bridge → 스위치와 투명 브리징

Bridge → 리피터·허브·브리지·스위치·라우터·게이트웨이

Bridged mode → VM 네트워크의 NAT·Bridge·Host-only

Bridge ID → 루트 브리지와 Bridge ID

Bridge Mode → Inline/Non-inline·Bridge/Router Mode·UTM/NGFW

Bridge Protocol Data Unit → BPDU

Bro → Snort·Suricata·Zeek

Broadcast → 유니캐스트·브로드캐스트·멀티캐스트·애니캐스트

Broadcast Domain → 충돌 도메인과 브로드캐스트 도메인

Broadcast Storm → 브로드캐스트 스톰

Broken Access Control → Broken Access Control·IDOR/BOLA

Broken Access Control·IDOR/BOLA

Brute Force → LAND·Teardrop·DDoS·Brute Force

BSS → 프로세스 메모리 구조

Burp Suite → Burp Suite·sqlmap·Hydra·John·Hashcat

Burp Suite·sqlmap·Hydra·John·Hashcat

Bus → 네트워크 토폴로지

bWAPP → Kali Linux·Metasploitable·DVWA·WebGoat·bee-box

byte → 비트·바이트·옥텟과 진법

C2 → Kill Chain·MITRE ATT&CK·C2

CA → TLS·Certificate·PKI·SSH

Cache → rsync·Mirror·CDN·Cache

cache → 메모리와 저장장치

Cacti → SNMP·OID·MIB·RRD

Cacti → Monitorix와 RRD

Call by Reference → 포인터·참조·인자 전달

Call by Value → 포인터·참조·인자 전달

Campus Area Network → PAN·LAN·CAN·MAN·WAN

Capture Filter → Wireshark·Npcap·tcpdump·PCAP 분석

Carrier Sense Multiple Access → CSMA/CD와 CSMA/CA

cat → cat·head·tail·grep·sed·tee·wget·curl·파이프

cat·head·tail·grep·sed·tee·wget·curl·파이프

C&C → Kill Chain·MITRE ATT&CK·C2

CDN → rsync·Mirror·CDN·Cache

Certificate → TLS·Certificate·PKI·SSH

CGI → CGI·FastCGI·PHP-FPM·WSGI·Worker

CGI·FastCGI·PHP-FPM·WSGI·Worker

Chain → Netfilter·Hook·Table·Chain·Rule

Chain of Custody → Incident Response·Evidence·Containment

CHAP → Serial·HDLC·PPP·LCP/NCP·PAP/CHAP

Checksum Incorrect → Offloading·Checksum 미완성·Promiscuous Mode

Checksum Offload → Offloading·Checksum 미완성·Promiscuous Mode

Chisel → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

chmod → Linux 권한·chmod · umask · Sticky bit

chown → Linux 권한·chmod · umask · Sticky bit

chroot → Virtual Hosting · SNI · Sandboxing

CIA Triad → CIA · 인증·인가·최소 권한

CIA · 인증·인가·최소 권한

CIDR → Subnet Mask · CIDR · VLSM · Summary

CIFS → FTP · SFTP · SMB/CIFS · Samba · NFS

Cisco IOS → GNS3 · 에뮬레이션·Cisco IOS CLI

class → 클래스·객체·인스턴스·getter/setter

Class A → IPv4 주소·Prefix · Network ID

Class B → IPv4 주소·Prefix · Network ID

Class C → IPv4 주소·Prefix · Network ID

CLI → Linux 커널·셸·터미널

Clickjacking → HTML Context · iframe · URL Encoding · CSP

Client → Client-Server · P2P · Keepalive · Load Balancing

Client-Server · P2P · Keepalive · Load Balancing

clone → ISO 이미지와 VM Snapshot

close → 파일 입출력·Context Manager · Race Condition

closure → 변수 Scope · LEGB · Closure

Cloud → On-premises · Cloud · IaC · Container

CMS → CMS와 DB 운영

CMS와 DB 운영

CNAME → DNS Record · TTL · 정방향/역방향 조회

coaxial → UTP · 광섬유·배선 규격

Code → 프로세스 메모리 구조

Collision Avoidance → CSMA/CD와 CSMA/CA

Collision Detection → CSMA/CD와 CSMA/CA

Collision Domain → 충돌 도메인과 브로드캐스트 도메인

Command and Control → Kill Chain · MITRE ATT&CK · C2

Command Injection → Command Injection · File Upload · SSRF · 보안 로깅

Command Injection · File Upload · SSRF · 보안 로깅

Community String → SNMP · OID · MIB · RRD

Compensating Control → 보안 기준과 실제 통제 검증

Compliance → Governance · Compliance

composition → 클래스·객체·인스턴스·getter/setter

conf.d → Virtual Hosting · SNI · Sandboxing

Confidentiality → CIA · 인증·인가·최소 권한

Configuration Revision → VTP

configure terminal → GNS3 · 에뮬레이션·Cisco IOS CLI

Congestion Control → TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

Connect Scan → Nmap · Port Scan · Service Detection

Conntrack → Netfilter · Hook · Table · Chain · Rule

Console → Linux 커널·셸·터미널

Console Line → Paramiko · Netmiko · Console · VTY

Container → On-premises · Cloud · IaC · Container

Containment → Incident Response · Evidence · Containment

Content Security Policy → HTML Context · iframe · URL Encoding · CSP

context manager → 파일 입출력·Context Manager · Race Condition

continue → 조건문·반복문·종료 조건

Continuous Verification → Zero Trust · Microsegmentation

Control Validation → 보안 기준과 실제 통제 검증

Convergence → Hop · Convergence · Routing Loop · Stub Network

Cookie → Cookie · Session · Session Hijacking

Cookie · Session · Session Hijacking

Copy-on-Write → Filesystem · Inode · Journaling · CoW

Correlation → SIEM · ELK · Wazuh

CORS → Same-Origin Policy · CORS · Origin

Count to Infinity → Count-to-Infinity · Split Horizon · Route Poisoning

Count-to-Infinity · Split Horizon · Route Poisoning

CoW → Filesystem · Inode · Journaling · CoW

CPU → CPU · 명령어·마이크로프로세싱

CPU Affinity → IRQ · RSS · Scheduler · Hyper-Threading

CPU · 명령어·마이크로프로세싱

CRC → 오류 제어·ARQ · Hamming Code

cross cable → UTP · 광섬유·배선 규격

Cross-Site Request Forgery → CSRF

Cross-Site Scripting → XSS

CSMA/CA → CSMA/CD와 CSMA/CA

CSMA/CD → CSMA/CD와 CSMA/CA

CSMA/CD와 CSMA/CA

CSP → HTML Context · iframe · URL Encoding · CSP

CSRF

CSRF Token → CSRF

Ctrl+C → 셸 파이프·리다이렉션·종료 상태

Ctrl+C → vi · Vim · nano와 Ctrl+C

curl → 패키지 관리자·Repository · upstream/downstream

curl → cat · head · tail · grep · sed · tee · wget · curl · 파이프

Cursor → DB · DBMS · RDBMS · SQL · Cursor

cursor → 파일 입출력·Context Manager · Race Condition

CVE → CVE · CWE · CVSS

CVE · CWE · CVSS

CVSS → CVE · CWE · CVSS

CWE → CVE · CWE · CVSS

Cyber Kill Chain → Kill Chain · MITRE ATT&CK · C2

Cylinder → 서버 부팅·UEFI · MBR · GPT · Partition

DAC → UTP · 광섬유·배선 규격

daemon → systemd · 서비스·daemon

DAI → Dynamic ARP Inspection

DAQ → DAQ · PCAP · Packet Capture

DAQ · PCAP · Packet Capture

Data → 프로세스 메모리 구조

Data Acquisition → DAQ · PCAP · Packet Capture

Database → DB · DBMS · RDBMS · SQL · Cursor

data type → Python 이름·객체·타입

DB · DBMS · RDBMS · SQL · Cursor

DBMS → DB · DBMS · RDBMS · SQL · Cursor

DB 계정·Host · Bind Address

DDoS → LAND · Teardrop · DDoS · Brute Force

Decapsulation → 캡슐화와 역캡슐화

Default Deny → CIA · 인증·인가·최소 권한

Default Gateway → Routing Table · Longest Prefix Match · Gateway

Default Route → Static · Dynamic Routing · Metric · AD · ECMP

Defense in Depth → 자산·위협·취약점·위험

DEI → IEEE 802.1Q (dot1q)

Deny → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Dependency Chain → End-to-End 요청 추적

Descriptor → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

Designated Port → STP 포트 역할

DHCP

DHCP Relay → DHCP

DHCP Snooping

DHCP 스누핑 → DHCP Snooping

dig → BIND · named · DNSSEC · dig

Digital Signature → 대칭키·비대칭키·Hash · Digital Signature

Dijkstra → OSPF · Link State · Dijkstra

Disk Quota → Mount · fstab · LVM · Quota

Display Filter → Wireshark · Npcap · tcpdump · PCAP 분석

Distance Vector → RIP와 Distance Vector

Distribute List → RIP Timer · Offset List · Distribute List

DLCI → Frame Relay · DLCI · PVC · Subinterface

DNAT → NAT · PAT · Masquerade · Floating IP

DNAT → Stateful Firewall · NAT Rule · Policy 검증

dnf → 패키지 관리자·Repository · upstream/downstream

DNS → 주요 응용 프로토콜과 Well-known Port

DNS · Domain · FQDN · Zone

DNSenum → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

DNSRecon → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

DNS Record · TTL · 정방향/역방향 조회

DNSSEC → BIND · named · DNSSEC · dig

DNS TTL → DNS Record · TTL · 정방향/역방향 조회

Docker → On-premises · Cloud · IaC · Container

Docker Lab → Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

DOM → HTML Context · iframe · URL Encoding · CSP

Domain → DNS · Domain · FQDN · Zone

Domain Name System → DNS · Domain · FQDN · Zone

DOM XSS → XSS

DORA → DHCP

dot1q → IEEE 802.1Q (dot1q)

downstream → 패키지 관리자·Repository · upstream/downstream

DPDK → BPF · eBPF · AF_PACKET · DPDK

DR → OSPF · Link State · Dijkstra

DSCP → QoS

DTP

DUAL → EIGRP · DUAL · Successor · SIA

Dummy Hub → 리피터·허브·브리지·스위치·라우터·게이트웨이

DVWA → Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

Dynamic ARP Inspection

dynamic auto → DTP

dynamic desirable → DTP

Dynamic Host Configuration Protocol → DHCP

Dynamic NAT → NAT · PAT · Masquerade · Floating IP

Dynamic Route → Static · Dynamic Routing · Metric · AD · ECMP

Dynamic Secure MAC → Secure MAC과 Sticky 학습

Dynamic Trunking Protocol → DTP

dynamic typing → Python 이름·객체·타입

eBGP → BGP · Path Vector

eBPF → BPF · eBPF · AF_PACKET · DPDK

ECC → 대칭키·비대칭키·Hash · Digital Signature

echo cancellation → UTP · 광섬유·배선 규격

ECMP → Static · Dynamic Routing · Metric · AD · ECMP

EDR → EDR · XDR · SOAR · 능동 대응

EDR · XDR · SOAR · 능동 대응

EGP → IGP · EGP · AS

EIGRP → EIGRP · DUAL · Successor · SIA

EIGRP · DUAL · Successor · SIA

Elasticsearch → SIEM · ELK · Wazuh

ELK Stack → SIEM · ELK · Wazuh

enable → GNS3 · 에뮬레이션·Cisco IOS CLI

Encapsulation → 캡슐화와 역캡슐화

Encapsulation → OSI 7계층과 PDU

encapsulation → 클래스·객체·인스턴스·getter/setter

End-to-End Troubleshooting → End-to-End 요청 추적

End-to-End 요청 추적

Enforcing → SELinux

Eradication → Incident Response · Evidence · Containment

error control → 오류 제어·ARQ · Hamming Code

error-disabled → Port Security violation mode

ERSPAN → SPAN · Port Mirroring · TAP

ESTABLISHED → Stateful Firewall · NAT Rule · Policy 검증

EtherChannel

Ethernet

EtherType → EtherType · IP Protocol Number · Port Number

EtherType · IP Protocol Number · Port Number

Evidence Integrity → Incident Response · Evidence · Containment

Exception Management → 보안 기준과 실제 통제 검증

exit code → 셸 파이프·리다이렉션·종료 상태

exit condition → 조건문·반복문·종료 조건

Exploit → 자산·위협·취약점·위험

Exploit → Metasploit · Exploit · Payload · Session · Reverse Shell

Exploitation → 모의해킹·Rules of Engagement

ext4 → Filesystem · Inode · Journaling · CoW

Extended Partition → 서버 부팅·UEFI · MBR · GPT · Partition

Exterior → IGP · EGP · AS

Fail2ban → Fail2ban · SSH Brute Force · TCP Wrapper

Fail2ban · SSH Brute Force · TCP Wrapper

Fail-close → Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

Fail-open → Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

False Negative → IDS · IPS · HIDS · NIDS · 탐지 방식

False Positive → IDS · IPS · HIDS · NIDS · 탐지 방식

FastCGI → CGI · FastCGI · PHP-FPM · WSGI · Worker

FCS → 오류 제어·ARQ · Hamming Code

FD → EIGRP · DUAL · Successor · SIA

FDM → UTP · 광섬유·배선 규격

Feasible Distance → EIGRP · DUAL · Successor · SIA

Feasible Successor → EIGRP · DUAL · Successor · SIA

Fetch Decode Execute → CPU · 명령어·마이크로프로세싱

ffuf → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

FHS → Linux 파일시스템 계층과 "모든 것은 파일"

fiber → UTP · 광섬유·배선 규격

file descriptor → Linux 파일시스템 계층과 "모든 것은 파일"

Filesystem → Filesystem · Inode · Journaling · CoW

Filesystem · Inode · Journaling · CoW

File Upload Vulnerability → Command Injection · File Upload · SSRF · 보안 로깅

FIN → TCP 연결 설정과 종료

firewalld → iptables · nftables · firewalld · UFW

Firewall Rich Rule → Stateful Firewall · NAT Rule · Policy 검증

first-class object → 함수·메서드·인자·1급 객체

Floating IP → NAT · PAT · Masquerade · Floating IP

Floating Static → Static · Dynamic Routing · Metric · AD · ECMP

Flooding과 MAC 주소 표 에이징

flow → Snort/Suricata Rule · SID · rev · flow · PCRE

Flow Control → TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

for → 조건문·반복문·종료 조건

formatted string → Python 자료형과 가변성

FORWARD → Netfilter · Hook · Table · Chain · Rule

Forward DNS → DNS Record · TTL · 정방향/역방향 조회

Forwarding → STP 포트 상태와 RSTP

fping → ICMP · Ping · Traceroute · TTL

FQDN → DNS · Domain · FQDN · Zone

Fragmentation Attack → Snort/Suricata Rule · SID · rev · flow · PCRE

Fragmentation Attack → LAND · Teardrop · DDoS · Brute Force

Frame Relay → Frame Relay · DLCI · PVC · Subinterface

Frame Relay · DLCI · PVC · Subinterface

f-string → Python 자료형과 가변성

FTP → 주요 응용 프로토콜과 Well-known Port

FTP → FTP · SFTP · SMB/CIFS · Samba · NFS

FTPS → FTP · SFTP · SMB/CIFS · Samba · NFS

FTP · SFTP · SMB/CIFS · Samba · NFS

Full Duplex → 대역폭·지연·Duplex

Full Update → RIP와 Distance Vector

Full Virtualization → 가상화·Hypervisor · VM

function → 함수·메서드·인자·1급 객체

Fuzzing → Packet Injection · Scapy · Fuzzing

Gateway → 리피터·허브·브리지·스위치·라우터·게이트웨이

Gateway of Last Resort → Routing Table · Longest Prefix Match · Gateway

getter → 클래스·객체·인스턴스·getter/setter

global → 변수 Scope · LEGB · Closure

GNS3 → GNS3 · 에뮬레이션·Cisco IOS CLI

GNS3 Security Lab → 통합 보안·네트워크 Lab

GNS3 · 에뮬레이션·Cisco IOS CLI

Gobuster → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

Governance → Governance · Compliance

Governance · Compliance

GPT → 서버 부팅·UEFI · MBR · GPT · Partition

Gratuitous ARP → ARP

grep → cat · head · tail · grep · sed · tee · wget · curl · 파이프

GRO → Offloading · Checksum 미완성·Promiscuous Mode

GSO → Offloading · Checksum 미완성·Promiscuous Mode

GUI → Linux 커널·셸·터미널

Guideline → Governance · Compliance

Half Duplex → 대역폭·지연·Duplex

Hamming code → 오류 제어·ARQ · Hamming Code

Hard Limit → Mount · fstab · LVM · Quota

hard link → Hard Link와 Symbolic Link

Hard Link와 Symbolic Link

Hash → 대칭키·비대칭키·Hash · Digital Signature

Hashcat → Burp Suite · sqlmap · Hydra · John · Hashcat

Hash Function → Hash · Salt · Rainbow Table

Hash · Salt · Rainbow Table

HDD → Block Device · Local/Remote Storage · SCSI/NVMe

HDLC → Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

head → Log · rsyslog · Severity · Trap

head → cat · head · tail · grep · sed · tee · wget · curl · 파이프

Header → HTTP · Method · Status · Header

Heap → 프로세스 메모리 구조

Heuristic-based Detection → IDS · IPS · HIDS · NIDS · 탐지 방식

hexadecimal → 비트·바이트·옥텟과 진법

hidden file → bashrc · zshrc · 숨김 파일·명령 history

HIDS → IDS · IPS · HIDS · NIDS · 탐지 방식

Hold-down → Count-to-Infinity · Split Horizon · Route Poisoning

Hop → Hop · Convergence · Routing Loop · Stub Network

Hop · Convergence · Routing Loop · Stub Network

Hop Count → RIP와 Distance Vector

Hop Limit → IPv6

Horizontal Privilege Escalation → Broken Access Control · IDOR/BOLA

Host Header → Virtual Hosting · SNI · Sandboxing

Host ID → IPv4 주소·Prefix · Network ID

Host Isolation → EDR · XDR · SOAR · 능동 대응

Host Key → TLS · Certificate · PKI · SSH

Host-only → VM 네트워크의 NAT · Bridge · Host-only

hosts.allow → Fail2ban · SSH Brute Force · TCP Wrapper

hosts.deny → Fail2ban · SSH Brute Force · TCP Wrapper

Hot Swap → RAID

HTML Context · iframe · URL Encoding · CSP

HTTP → 주요 응용 프로토콜과 Well-known Port

HTTP → HTTP · Method · Status · Header

HTTP · Method · Status · Header

HttpOnly → Cookie · Session · Session Hijacking

HTTPS → 주요 응용 프로토콜과 Well-known Port

HTTPS → HTTP · Method · Status · Header

Hub → 리피터·허브·브리지·스위치·라우터·게이트웨이

Hub and Spoke → NBMA 환경과 RIP

Hydra → Burp Suite · sqlmap · Hydra · John · Hashcat

Hypercall → 가상화·Hypervisor · VM

Hyper-Threading → IRQ · RSS · Scheduler · Hyper-Threading

Hypervisor → 가상화·Hypervisor · VM

IANA → 사설 주소·공인 주소·IANA · ISP

iBGP → BGP · Path Vector

ICMP → ICMP · Ping · Traceroute · TTL

ICMP · Ping · Traceroute · TTL

ICMPv6 → IPv6

IDE → Block Device · Local/Remote Storage · SCSI/NVMe

IDOR → Broken Access Control · IDOR/BOLA

IDS → IDS · IPS · HIDS · NIDS · 탐지 방식

IDS · IPS · HIDS · NIDS · 탐지 방식

IDS Troubleshooting → L2 · Routing · IDS 장애 대응

IEEE → ISO · IEEE · IETF · RFC

IEEE 802.1Q (dot1q)

IEEE 802.3 → Ethernet

IETF → ISO · IEEE · IETF · RFC

if → 조건문·반복문·종료 조건

iframe → HTML Context · iframe · URL Encoding · CSP

IGP → IGP · EGP · AS

IGP · EGP · AS

IIS → LAMP · Web Server · WAS

iLO → Rack Server · Blade Server · BMC/iLO

immutable → Python 자료형과 가변성

Impact → 자산·위협·취약점·위험

Implicit Deny → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Inbound → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Incident Response → Incident Response · Evidence · Containment

Incident Response · Evidence · Containment

infinite loop → 조건문·반복문·종료 조건

Infrastructure as Code → On-premises · Cloud · IaC · Container

inheritance → 클래스·객체·인스턴스·getter/setter

init → systemd · 서비스·daemon

Inline IDS/IPS → Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

Inode → Filesystem · Inode · Journaling · CoW

Inode Exhaustion → Filesystem · Inode · Journaling · CoW

inode link count → Hard Link와 Symbolic Link

INPUT → Netfilter · Hook · Table · Chain · Rule

instance → 클래스·객체·인스턴스·getter/setter

Integrated Lab → 통합 보안·네트워크 Lab

Integrity → CIA · 인증·인가·최소 권한

Interior → IGP · EGP · AS

Internet Standard → ISO · IEEE · IETF · RFC

Interrupt → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

Inter-VLAN routing

IOSv → GNS3 · 에뮬레이션·Cisco IOS CLI

IOU → GNS3 · 에뮬레이션·Cisco IOS CLI

IP address → IPv4 주소·Prefix · Network ID

IP-based Virtual Hosting → Virtual Hosting · SNI · Sandboxing

ipconfig /flushdns → BIND · named · DNSSEC · dig

IPS → IDS · IPS · HIDS · NIDS · 탐지 방식

iptables → iptables · nftables · firewalld · UFW

iptables · nftables · firewalld · UFW

IPv4 → IPv4 주소·Prefix · Network ID

IPv4 주소·Prefix · Network ID

IPv6

IRQ → IRQ · RSS · Scheduler · Hyper-Threading

IRQ · RSS · Scheduler · Hyper-Threading

iSCSI → Block Device · Local/Remote Storage · SCSI/NVMe

ISO → ISO · IEEE · IETF · RFC

ISO · IEEE · IETF · RFC

ISO image → ISO 이미지와 VM Snapshot

ISO 이미지와 VM Snapshot

ISP → 사설 주소·공인 주소·IANA · ISP

Jail → Fail2ban · SSH Brute Force · TCP Wrapper

JavaScript Context → HTML Context · iframe · URL Encoding · CSP

JFS → Filesystem · Inode · Journaling · CoW

John the Ripper → Burp Suite · sqlmap · Hydra · John · Hashcat

Joomla → CMS와 DB 운영

Journaling → Filesystem · Inode · Journaling · CoW

JSP → LAMP · Web Server · WAS

Jupyter Notebook → 실행 인자와 Python 가상 환경

Kali Linux → Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

Keep Alive → Client-Server · P2P · Keepalive · Load Balancing

Kernel → Linux 커널·셸·터미널

Kernel Network Stack → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

Key Size → 대칭키·비대칭키·Hash · Digital Signature

Kibana → SIEM · ELK · Wazuh

Kill Chain · MITRE ATT&CK · C2

KISA → OWASP · KISA · 주요정보통신기반시설 가이드

K values → EIGRP · DUAL · Successor · SIA

kwargs → 함수·메서드·인자·1급 객체

L2 Loop Response → L2 · Routing · IDS 장애 대응

L2 · Routing · IDS 장애 대응

L2 루프와 루핑

L3 Switch → Inter-VLAN routing

L3 Switch → 리피터·허브·브리지·스위치·라우터·게이트웨이

LACP → EtherChannel

LAG → EtherChannel

lambda → 함수·메서드·인자·1급 객체

LAMP → LAMP · Web Server · WAS

LAMP · Web Server · WAS

LAND Attack → LAND · Teardrop · DDoS · Brute Force

LAND · Teardrop · DDoS · Brute Force

LAN과 VLAN → VLAN

Latency → 대역폭·지연·Duplex

Layer 2 Loop → L2 루프와 루핑

Layered Diagnosis → 계층별 트러블슈팅

LCP → Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

Learning → STP 포트 상태와 RSTP

Least Privilege → CIA · 인증·인가·최소 권한

LEGB → 변수 Scope · LEGB · Closure

libpcap → DAQ · PCAP · Packet Capture

Line VTY → Paramiko · Netmiko · Console · VTY

Link Aggregation → EtherChannel

Link State → OSPF · Link State · Dijkstra

LinPEAS → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

Linux → Linux 커널·셸·터미널

Linux 권한·chmod · umask · Sticky bit

Linux 커널·셸·터미널

Linux 파일시스템 계층과 “모든 것은 파일”

list → Python 자료형과 가변성

listen → Port · Socket · 5-tuple

Listening → STP 포트 상태와 RSTP

Load Balancer → Client-Server · P2P · Keepalive · Load Balancing

Local Area Network → PAN · LAN · CAN · MAN · WAN

localhost → Loopback과 127.0.0.1

Local SPAN → SPAN · Port Mirroring · TAP

logging trap → Log · rsyslog · Severity · Trap

logrotate → Logrotate · NTP · Serialization

Logrotate · NTP · Serialization

Log Rotation → Logrotate · NTP · Serialization

Log · rsyslog · Severity · Trap

Logstash → SIEM · ELK · Wazuh

Longest Prefix Match → Routing Table · Longest Prefix Match · Gateway

Loopback Interface → Loopback과 127.0.0.1

Loopback과 127.0.0.1

Loop Guard → STP 보호 기능

Looping → L2 루프와 루핑

lost+found → Filesystem · Inode · Journaling · CoW

LRO → Offloading · Checksum 미완성·Promiscuous Mode

LSA → OSPF · Link State · Dijkstra

LSDB → OSPF · Link State · Dijkstra

Lua → Snort · Suricata · Zeek

LV → Mount · fstab · LVM · Quota

LVM → Mount · fstab · LVM · Quota

MAC → SELinux

mac-address sticky → Secure MAC과 Sticky 학습

MAC Flapping → MAC 플래핑

MAC Move → MAC 플래핑

MAC Table Aging → Flooding과 MAC 주소 표 에이징

MAC 주소

MAC 플래핑

Mainframe → Mainframe · Workstation · VDI

Mainframe · Workstation · VDI

Major Network → IPv4 주소·Prefix · Network ID

Maltego → OSINT · WHOIS · Shodan · Maltego

Mandatory Access Control → SELinux

Man in the Middle → Sniffing · Spoofing · MITM

MariaDB → DB 계정·Host · Bind Address

Masquerade → NAT · PAT · Masquerade · Floating IP

Masquerade → Stateful Firewall · NAT Rule · Policy 검증

MBR → 서버 부팅·UEFI · MBR · GPT · Partition

mdadm → Mount · fstab · LVM · Quota

Media Access Control Address → MAC 주소

Memory allocation → 메모리와 저장장치

Mesh → 네트워크 토폴로지

metadisk → Mount · fstab · LVM · Quota

Metasploit → Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

Metasploitable → Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

Metasploit · Exploit · Payload · Session · Reverse Shell

Metasploit Framework → Metasploit · Exploit · Payload · Session · Reverse Shell

Method → HTTP · Method · Status · Header

method → 함수·메서드·인자·1급 객체

Metric → Static · Dynamic Routing · Metric · AD · ECMP

Metropolitan Area Network → PAN · LAN · CAN · MAN · WAN

MIB → SNMP · OID · MIB · RRD

Microprocessor → CPU · 명령어·마이크로프로세싱

Microsegmentation → Zero Trust · Microsegmentation

Mimikatz → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

Mirror Domain → rsync · Mirror · CDN · Cache

Mirroring → RAID

MITM → Sniffing · Spoofing · MITM

MITRE ATT&CK → Kill Chain · MITRE ATT&CK · C2

MMU → 가상 메모리와 Swap

Monitorix → Monitorix와 RRD

Monitorix와 RRD

mount → Mount · fstab · LVM · Quota

mount.cifs → FTP · SFTP · SMB/CIFS · Samba · NFS

Mount · fstab · LVM · Quota

MRTG → SNMP · OID · MIB · RRD

MRTG → Monitorix와 RRD

Multicast → 유니캐스트·브로드캐스트·멀티캐스트·애니캐스트

Multipoint → Frame Relay · DLCI · PVC · Subinterface

mutable → Python 자료형과 가변성

MX → DNS Record · TTL · 정방향/역방향 조회

MySQL → 주요 응용 프로토콜과 Well-known Port

MySQL → DB 계정·Host · Bind Address

Name-based Virtual Hosting → Virtual Hosting · SNI · Sandboxing

named → BIND · named · DNSSEC · dig

nano → vi · Vim · nano와 Ctrl+C

NAPI → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

NAPT → NAT · PAT · Masquerade · Floating IP

NAS → Block Device · Local/Remote Storage · SCSI/NVMe

Native VLAN

Native VLAN mismatch → Native VLAN

NAT mode → VM 네트워크의 NAT · Bridge · Host-only

NAT · PAT · Masquerade · Floating IP

NBMA → NBMA 환경과 RIP

NBMA 환경과 RIP

NCP → Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

Neighbor Discovery → IPv6

neighbor table → ARP

Nessus → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

nested code → 조건문·반복문·종료 조건

NetExec → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

Netfilter → Netfilter · Hook · Table · Chain · Rule

Netfilter · Hook · Table · Chain · Rule

Net ID → IPv4 주소·Prefix · Network ID

Netmask → Subnet Mask · CIDR · VLSM · Summary

Netmiko → Paramiko · Netmiko · Console · VTY

Network Address Translation → NAT · PAT · Masquerade · Floating IP

Network Interface → OSI 7계층과 PDU

Network Interface Controller → NIC

Network Security Monitoring → NSM · Security Onion · Sguil/Squert/Barnyard2

Network TAP → SPAN · Port Mirroring · TAP

Network Topology → 네트워크 토폴로지

Next Header → EtherType · IP Protocol Number · Port Number

Next Hop → Routing Table · Longest Prefix Match · Gateway

NFS → Block Device · Local/Remote Storage · SCSI/NVMe

NFS → FTP · SFTP · SMB/CIFS · Samba · NFS

nft → iptables · nftables · firewalld · UFW

nftables → iptables · nftables · firewalld · UFW

NGFW → Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

Nginx → LAMP · Web Server · WAS

NI → OSI 7계층과 PDU

NIC

NIC Driver → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

NIDS → IDS · IPS · HIDS · NIDS · 탐지 방식

Nikto → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

Nmap → Nmap · Port Scan · Service Detection

Nmap · Port Scan · Service Detection

Nmap Scripting Engine → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

nmcli → DHCP

Non-Broadcast Multiple Access → NBMA 환경과 RIP

Non-inline → Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

nonlocal → 변수 Scope · LEGB · Closure

no shutdown → GNS3 · 에뮬레이션·Cisco IOS CLI

NoSQL → DB · DBMS · RDBMS · SQL · Cursor

Npcap → Wireshark · Npcap · tcpdump · PCAP 분석

NS → DNS Record · TTL · 정방향/역방향 조회

NSE → NSE · DNS enumeration · Web discovery · Vulnerability Scanner

NSE · DNS enumeration · Web discovery · Vulnerability Scanner

nslookup → BIND · named · DNSSEC · dig

NSM → NSM · Security Onion · Sguil/Squert/Barnyard2

NSM · Security Onion · Sguil/Squert/Barnyard2

NTP → 주요 응용 프로토콜과 Well-known Port

NTP → Logrotate · NTP · Serialization

NVMe → Block Device · Local/Remote Storage · SCSI/NVMe

object → Python 이름·객체·타입

object → 클래스·객체·인스턴스·getter/setter

object sharing → 포인터·참조·인자 전달

octet → 비트·바이트·옥텟과 진법

Offloading · Checksum 미완성·Promiscuous Mode

Offset List → RIP Timer · Offset List · Distribute List

OID → SNMP · OID · MIB · RRD

One-click → Cookie · Session · Session Hijacking

On-premises → On-premises · Cloud · IaC · Container

On-premises · Cloud · IaC · Container

open → 파일 입출력·Context Manager · Race Condition

Open Source Intelligence → OSINT · WHOIS · Shodan · Maltego

OpenStack → On-premises · Cloud · IaC · Container

operand → CPU · 명령어·마이크로프로세싱

operator → CPU · 명령어·마이크로프로세싱

Origin → Same-Origin Policy · CORS · Origin

Origin Check → CSRF

OS Detection → Nmap · Port Scan · Service Detection

OSI 7 Layer → OSI 7계층과 PDU

OSI 7계층과 PDU

OSINT → OSINT · WHOIS · Shodan · Maltego

OSINT · WHOIS · Shodan · Maltego

OSPF → OSPF · Link State · Dijkstra

OSPF · Link State · Dijkstra

OUI → MAC 주소

Outbound → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Out-of-band Management → Rack Server · Blade Server · BMC/iLO

OUTPUT → Netfilter · Hook · Table · Chain · Rule

Output Encoding → XSS

OWASP → OWASP · KISA · 주요정보통신기반시설 가이드

OWASP · KISA · 주요정보통신기반시설 가이드

OWASP Top 10 → OWASP · KISA · 주요정보통신기반시설 가이드

P2P → Client-Server · P2P · Keepalive · Load Balancing

Packet Builder → Packet Injection · Scapy · Fuzzing

Packet Injection → Packet Injection · Scapy · Fuzzing

Packet Injection · Scapy · Fuzzing

Packet Path → 계층별 트러블슈팅

Page Fault → 가상 메모리와 Swap

Page Table → 가상 메모리와 Swap

PAgP → EtherChannel

PAN · LAN · CAN · MAN · WAN

PAP → Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

parameter → 함수·메서드·인자·1급 객체

Parameterized Query → SQL Injection · Blind SQL Injection

Paramiko → Paramiko · Netmiko · Console · VTY

Paramiko · Netmiko · Console · VTY

Paravirtualization → 가상화·Hypervisor · VM

Parity → RAID

pass → 조건문·반복문·종료 조건

Passive Interface → RIP Timer · Offset List · Distribute List

Password Hashing → Hash · Salt · Rainbow Table

PAT → NAT · PAT · Masquerade · Floating IP

Path Vector → BGP · Path Vector

Payload → Metasploit · Exploit · Payload · Session · Reverse Shell

PBKDF2 → Hash · Salt · Rainbow Table

PCAP → DAQ · PCAP · Packet Capture

PCAP → Wireshark · Npcap · tcpdump · PCAP 분석

pcapng → DAQ · PCAP · Packet Capture

PCP → IEEE 802.1Q (dot1q)

PCP → QoS

PCRE → Snort/Suricata Rule · SID · rev · flow · PCRE

PDU → 캡슐화와 역캡슐화

PDU → OSI 7계층과 PDU

Penetration Testing → 모의해킹·Rules of Engagement

Pentest → 모의해킹·Rules of Engagement

Pentest Lab → 통합 보안·네트워크 Lab

Percent Encoding → HTML Context · iframe · URL Encoding · CSP

permission → Linux 권한·chmod · umask · Sticky bit

Permissive → SELinux

Permit → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Personal Area Network → PAN · LAN · CAN · MAN · WAN

PHP-FPM → CGI · FastCGI · PHP-FPM · WSGI · Worker

phpMyAdmin → DB 계정·Host · Bind Address

PID → 프로그램·프로세스·PID

ping → ICMP · Ping · Traceroute · TTL

pipe → 셸 파이프·리다이렉션·종료 상태

pipe → cat · head · tail · grep · sed · tee · wget · curl · 파이프

PKI → TLS · Certificate · PKI · SSH

Playbook → EDR · XDR · SOAR · 능동 대응

Pointer → 포인터·참조·인자 전달

Point-in-time Recovery → CMS와 DB 운영

Point-to-point → Frame Relay · DLCI · PVC · Subinterface

Poison Reverse → Count-to-Infinity · Split Horizon · Route Poisoning

Policing → QoS

Policy → Governance · Compliance

Polling → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

port → Port · Socket · 5-tuple

Port-based Virtual Hosting → Virtual Hosting · SNI · Sandboxing

Port-Channel → EtherChannel

PortFast → STP 보호 기능

Port Mirroring → SPAN · Port Mirroring · TAP

Port Number → EtherType · IP Protocol Number · Port Number

Port Scan → Nmap · Port Scan · Service Detection

Port Security

Port Security violation mode

Port · Socket · 5-tuple

Post Exploitation → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

POSTROUTING → Netfilter · Hook · Table · Chain · Rule

PPID → 프로그램·프로세스·PID

PPP → Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

PPPoE → Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

Preflight → Same-Origin Policy · CORS · Origin

Prepared Statement → SQL Injection · Blind SQL Injection

PREROUTING → Netfilter · Hook · Table · Chain · Rule

Primary Partition → 서버 부팅·UEFI · MBR · GPT · Partition

Private IP → 사설 주소·공인 주소·IANA · ISP

Privilege → DB 계정·Host · Bind Address

Procedure → Kill Chain · MITRE ATT&CK · C2

Procedure → Governance · Compliance

Process → 프로그램·프로세스·PID

Program → 프로그램·프로세스·PID

Promiscuous Mode → Offloading · Checksum 미완성·Promiscuous Mode

protect → Port Security violation mode

Protocol Number → EtherType · IP Protocol Number · Port Number

Proxy ARP → ARP

PTR → DNS Record · TTL · 정방향/역방향 조회

Public IP → 사설 주소·공인 주소·IANA · ISP

PV → Mount · fstab · LVM · Quota

PVC → Frame Relay · DLCI · PVC · Subinterface

PyInstaller → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

Python 이름·객체·타입

Python 자료형과 가변성

QEMU → GNS3 · 에뮬레이션·Cisco IOS CLI

QoS

Quality of Service → QoS

QUIC → UDP · QUIC · SCTP

race condition → 파일 입출력·Context Manager · Race Condition

Rack Server · Blade Server · BMC/iLO

RAID

RAID 0 → RAID

RAID 01 → RAID

RAID 1 → RAID

RAID 10 → RAID

RAID 5 → RAID

RAID 6 → RAID

Rainbow Table → Hash · Salt · Rainbow Table

RAM → 메모리와 저장장치

range → 조건문·반복문·종료 조건

rc file → bashrc · zshrc · 숨김 파일·명령 history

RD → EIGRP · DUAL · Successor · SIA

RDAP → OSINT · WHOIS · Shodan · Maltego

RDBMS → DB · DBMS · RDBMS · SQL · Cursor

RDoS → LAND · Teardrop · DDoS · Brute Force

read → 파일 입출력·Context Manager · Race Condition

Reconnaissance → 모의해킹·Rules of Engagement

Recovery → Incident Response · Evidence · Containment

Recursive Resolver → DNS · Domain · FQDN · Zone

redirection → 셸 파이프·리다이렉션·종료 상태

Reference → 포인터·참조·인자 전달

reference → Python 이름·객체·타입

Reflected XSS → XSS

RELATED → Stateful Firewall · NAT Rule · Policy 검증

Repeater → 리피터·허브·브리지·스위치·라우터·게이트웨이

repo → 패키지 관리자·Repository · upstream/downstream

Reported Distance → EIGRP · DUAL · Successor · SIA

repository → 패키지 관리자·Repository · upstream/downstream

Request → HTTP · Method · Status · Header

Request for Comments → ISO · IEEE · IETF · RFC

resource pool → On-premises · Cloud · IaC · Container

Responder → Post-exploitation · LinPEAS · Mimikatz · Responder · NetExec · Chisel · Autopsy

Response → HTTP · Method · Status · Header

restrict → Port Security violation mode

Retest → 모의해킹·Rules of Engagement

Retransmission → TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

return 0 → 셸 파이프·리다이렉션·종료 상태

rev → Snort/Suricata Rule · SID · rev · flow · PCRE

Reverse DNS → DNS Record · TTL · 정방향/역방향 조회

Reverse Shell → Metasploit · Exploit · Payload · Session · Reverse Shell

RFC → ISO · IEEE · IETF · RFC

RFC1918 → 사설 주소·공인 주소·IANA · ISP

Rich Rule → iptables · nftables · firewalld · UFW

Ring → 네트워크 토폴로지

RIP → RIP와 Distance Vector

RIP Timer · Offset List · Distribute List

RIP Timers → RIP Timer · Offset List · Distribute List

RIPv1 → RIP와 Distance Vector

RIPv2 → RIP와 Distance Vector

RIP와 Distance Vector

Risk → 자산·위협·취약점·위험

Risk Appetite → Governance · Compliance

Root Bridge → 루트 브리지와 Bridge ID

Root DNS → DNS · Domain · FQDN · Zone

Root Election → 루트 브리지와 Bridge ID

Root Guard → STP 보호 기능

Root Port → STP 포트 역할

Round Robin → Client-Server · P2P · Keepalive · Load Balancing

Route Aggregation → Subnet Mask · CIDR · VLSM · Summary

Route Poisoning → Count-to-Infinity · Split Horizon · Route Poisoning

Router → 리피터·허브·브리지·스위치·라우터·게이트웨이

Router Mode → Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

Router-on-a-stick → Inter-VLAN routing

Route Summary → Subnet Mask · CIDR · VLSM · Summary

Routing Loop → Hop · Convergence · Routing Loop · Stub Network

Routing Table → Routing Table · Longest Prefix Match · Gateway

Routing Table · Longest Prefix Match · Gateway

Routing Troubleshooting → L2 · Routing · IDS 장애 대응

RPKI → BGP · Path Vector

RRA → SNMP · OID · MIB · RRD

RRD → SNMP · OID · MIB · RRD

RRD → Monitorix와 RRD

RRDTool → SNMP · OID · MIB · RRD

RRDtool → Monitorix와 RRD

RSA → 대칭키·비대칭키·Hash · Digital Signature

RSPAN → SPAN · Port Mirroring · TAP

RSS → IRQ · RSS · Scheduler · Hyper-Threading

RST → TCP 연결 설정과 종료

RSTP → STP 포트 상태와 RSTP

rsync → rsync · Mirror · CDN · Cache

rsync · Mirror · CDN · Cache

rsyslogd → Log · rsyslog · Severity · Trap

Rule → Netfilter · Hook · Table · Chain · Rule

Ruleset → Snort/Suricata Rule · SID · rev · flow · PCRE

Rules of Engagement → 모의해킹·Rules of Engagement

running-config → GNS3 · 에뮬레이션·Cisco IOS CLI

rwx → Linux 권한·chmod · umask · Sticky bit

RX Buffer → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

RX Queue → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

RX Ring → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

SACK → TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

Salt → Hash · Salt · Rainbow Table

Samba → FTP · SFTP · SMB/CIFS · Samba · NFS

Same-Origin Policy → Same-Origin Policy · CORS · Origin

Same-Origin Policy · CORS · Origin

SameSite → Cookie · Session · Session Hijacking

SameSite → CSRF

SAN → Block Device · Local/Remote Storage · SCSI/NVMe

Sandbox → Virtual Hosting · SNI · Sandboxing

Sanitization → XSS

SAS → Block Device · Local/Remote Storage · SCSI/NVMe

Scapy → Packet Injection · Scapy · Fuzzing

Scheduler → IRQ · RSS · Scheduler · Hyper-Threading

Scheduling → QoS

Scope → 모의해킹·Rules of Engagement

scope → 변수 Scope · LEGB · Closure

SCSI → Block Device · Local/Remote Storage · SCSI/NVMe

SCTP → UDP · QUIC · SCTP

Sector → 서버 부팅·UEFI · MBR · GPT · Partition

Secure → Cookie · Session · Session Hijacking

Secure MAC Address → Secure MAC과 Sticky 학습

Secure MAC과 Sticky 학습

Security Baseline → 보안 기준과 실제 통제 검증

Security-Enhanced Linux → SELinux

Security Logging and Monitoring Failures → Command Injection · File Upload · SSRF · 보안 로깅

Security Onion → NSM · Security Onion · Sguil/Squert/Barnyard2

sed → cat · head · tail · grep · sed · tee · wget · curl · 파이프

SELinux

Sequence Number → TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

Serial Interface → Serial · HDLC · PPP · LCP/NCP · PAP/CHAP

Serialization → Logrotate · NTP · Serialization

Server → Client-Server · P2P · Keepalive · Load Balancing

Server Rack → Rack Server · Blade Server · BMC/iLO

Server-side Script → LAMP · Web Server · WAS

service → systemd · 서비스·daemon

Service Detection → Nmap · Port Scan · Service Detection

Service Interface → OSI 7계층과 PDU

Session → Cookie · Session · Session Hijacking

Session → Metasploit · Exploit · Payload · Session · Reverse Shell

Session Hijacking → Cookie · Session · Session Hijacking

Session ID → Cookie · Session · Session Hijacking

Session Sidejacking → Sniffing · Spoofing · MITM

SET → Burp Suite · sqlmap · Hydra · John · Hashcat

setter → 클래스·객체·인스턴스·getter/setter

SFTP → FTP · SFTP · SMB/CIFS · Samba · NFS

Sguil → NSM · Security Onion · Sguil/Squert/Barnyard2

Shaping → QoS

Shell → Linux 커널·셸·터미널

shell history → bashrc · zshrc · 숨김 파일·명령 history

Shodan → OSINT · WHOIS · Shodan · Maltego

shutdown → Port Security violation mode

SI → OSI 7계층과 PDU

SIA → EIGRP · DUAL · Successor · SIA

SID → Snort/Suricata Rule · SID · rev · flow · PCRE

Sidejacking → Cookie · Session · Session Hijacking

SIEM → SIEM · ELK · Wazuh

SIEM · ELK · Wazuh

SIGINT → 셸 파이프·리다이렉션·종료 상태

SIGINT → vi · Vim · nano와 Ctrl+C

Signature-based Detection → IDS · IPS · HIDS · NIDS · 탐지 방식

Simplex → 대역폭·지연·Duplex

skb → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

sk_buff → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

Sliding Window → TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

SMB → Block Device · Local/Remote Storage · SCSI/NVMe

SMB → FTP · SFTP · SMB/CIFS · Samba · NFS

SMT → IRQ · RSS · Scheduler · Hyper-Threading

SMTP → 주요 응용 프로토콜과 Well-known Port

snapshot → ISO 이미지와 VM Snapshot

SNAT → NAT · PAT · Masquerade · Floating IP

SNAT → Stateful Firewall · NAT Rule · Policy 검증

SNI → Virtual Hosting · SNI · Sandboxing

SNI → TLS · Certificate · PKI · SSH

Sniffing → Sniffing · Spoofing · MITM

Sniffing · Spoofing · MITM

SNMP → 주요 응용 프로토콜과 Well-known Port

SNMP → SNMP · OID · MIB · RRD

SNMP Agent → SNMP · OID · MIB · RRD

SNMP Manager → SNMP · OID · MIB · RRD

SNMP · OID · MIB · RRD

SNMPv1 → SNMP · OID · MIB · RRD

SNMPv2c → SNMP · OID · MIB · RRD

SNMPv3 → SNMP · OID · MIB · RRD

Snooping Binding Database → DHCP Snooping

Snort → Snort · Suricata · Zeek

Snort Rule → Snort/Suricata Rule · SID · rev · flow · PCRE

Snort/Suricata Rule · SID · rev · flow · PCRE

Snort · Suricata · Zeek

SOA → DNS Record · TTL · 정방향/역방향 조회

SOAR → EDR · XDR · SOAR · 능동 대응

socket → Port · Socket · 5-tuple

Socket Receive Buffer → NIC RX Queue · Descriptor · NAPI · sk_buff · Socket Buffer

Soft Limit → Mount · fstab · LVM · Quota

SOP → Same-Origin Policy · CORS · Origin

SPAN → SPAN · Port Mirroring · TAP

Spanning Tree Protocol → STP

SPAN · Port Mirroring · TAP

SPF → OSPF · Link State · Dijkstra

Split Horizon → Count-to-Infinity · Split Horizon · Route Poisoning

Spoofing → Sniffing · Spoofing · MITM

SPU → Rack Server · Blade Server · BMC/iLO

SQL → DB · DBMS · RDBMS · SQL · Cursor

SQLi → SQL Injection · Blind SQL Injection

SQL Injection → SQL Injection · Blind SQL Injection

SQL Injection · Blind SQL Injection

sqlmap → Burp Suite · sqlmap · Hydra · John · Hashcat

Squert → NSM · Security Onion · Sguil/Squert/Barnyard2

SRV → DNS Record · TTL · 정방향/역방향 조회

SSD → Block Device · Local/Remote Storage · SCSI/NVMe

SSH → 주요 응용 프로토콜과 Well-known Port

SSH → TLS · Certificate · PKI · SSH

SSH automation → Paramiko · Netmiko · Console · VTY

SSH Brute Force → Fail2ban · SSH Brute Force · TCP Wrapper

SSL → TLS · Certificate · PKI · SSH

SSRF → Command Injection · File Upload · SSRF · 보안 로깅

Stack → 프로세스 메모리 구조

stack frame → 프로세스 메모리 구조

Standard → Governance · Compliance

Star → 네트워크 토폴로지

startup-config → GNS3 · 에뮬레이션·Cisco IOS CLI

Stateful Firewall → Stateful Firewall · NAT Rule · Policy 검증

Stateful Firewall · NAT Rule · Policy 검증

Static · Dynamic Routing · Metric · AD · ECMP

Static NAT → NAT · PAT · Masquerade · Floating IP

Static Route → Static · Dynamic Routing · Metric · AD · ECMP

Static Secure MAC → Secure MAC과 Sticky 학습

Status Code → HTTP · Method · Status · Header

sticky bit → Linux 권한·chmod · umask · Sticky bit

Sticky Port → Secure MAC과 Sticky 학습

Storage → 메모리와 저장장치

Stored XSS → XSS

Storm Control → 브로드캐스트 스톰

STP

STP 보호 기능

STP 포트 상태와 RSTP

STP 포트 역할

str → Python 자료형과 가변성

Striping → RAID

Stub Network → Hop · Convergence · Routing Loop · Stub Network

Stuck in Active → EIGRP · DUAL · Successor · SIA

Subinterface → Inter-VLAN routing

Subinterface → Frame Relay · DLCI · PVC · Subinterface

Subnet Mask · CIDR · VLSM · Summary

Subnetting → Subnet Mask · CIDR · VLSM · Summary

sub-TLV → TLV

Successor → EIGRP · DUAL · Successor · SIA

sudo → sudo · Linux 계정·Service User

sudo · Linux 계정·Service User

Superior BPDU → BPDU

Supernetting → Subnet Mask · CIDR · VLSM · Summary

Superscalar → IRQ · RSS · Scheduler · Hyper-Threading

Suricata → Snort · Suricata · Zeek

Suricata Rule → Snort/Suricata Rule · SID · rev · flow · PCRE

SVC → Frame Relay · DLCI · PVC · Subinterface

SVI → Inter-VLAN routing

swap → 가상 메모리와 Swap

Switch → 스위치와 투명 브리징

Switch → 리피터·허브·브리지·스위치·라우터·게이트웨이

Switched Port Analyzer → SPAN · Port Mirroring · TAP

switchport port-security → Port Security

symbolic link → Hard Link와 Symbolic Link

symlink → Hard Link와 Symbolic Link

Symmetric Cryptography → 대칭키·비대칭키·Hash · Digital Signature

SYN → TCP 연결 설정과 종료

SYN Scan → Nmap · Port Scan · Service Detection

Syslog Severity → Log · rsyslog · Severity · Trap

systemctl → systemd · 서비스·daemon

systemd → systemd · 서비스·daemon

systemd · 서비스·daemon

System Group → sudo · Linux 계정·Service User

System User → sudo · Linux 계정·Service User

system user → Virtual Hosting · SNI · Sandboxing

T568A → UTP · 광섬유·배선 규격

T568B → UTP · 광섬유·배선 규격

Table → Netfilter · Hook · Table · Chain · Rule

Tactic → Kill Chain · MITRE ATT&CK · C2

tail → Log · rsyslog · Severity · Trap

tail → cat · head · tail · grep · sed · tee · wget · curl · 파이프

TCP → TCP 연결 설정과 종료

tcpdump → DAQ · PCAP · Packet Capture

tcpdump → Wireshark · Npcap · tcpdump · PCAP 분석

TCP Follow → Snort/Suricata Rule · SID · rev · flow · PCRE

TCP Wrapper → Fail2ban · SSH Brute Force · TCP Wrapper

TCP 신뢰성·Sliding Window · 흐름/혼잡 제어

TCP 연결 설정과 종료

Teardrop → Snort/Suricata Rule · SID · rev · flow · PCRE

Teardrop → LAND · Teardrop · DDoS · Brute Force

Technique → Kill Chain · MITRE ATT&CK · C2

tee → cat · head · tail · grep · sed · tee · wget · curl · 파이프

Telnet → 주요 응용 프로토콜과 Well-known Port

Telnet → Paramiko · Netmiko · Console · VTY

Terminal → Linux 커널·셸·터미널

Terraform → On-premises · Cloud · IaC · Container

Text → 프로세스 메모리 구조

Threat → 자산·위협·취약점·위험

Throughput → 대역폭·지연·Duplex

Time Exceeded → ICMP · Ping · Traceroute · TTL

Time Synchronization → Logrotate · NTP · Serialization

TIME_WAIT → TCP 연결 설정과 종료

TLD → DNS · Domain · FQDN · Zone

TLS → TLS · Certificate · PKI · SSH

TLS · Certificate · PKI · SSH

TLV

Token Passing → 네트워크 토폴로지

Tomcat → LAMP · Web Server · WAS

TPID → IEEE 802.1Q (dot1q)

traceroute → ICMP · Ping · Traceroute · TTL

Transceiver → 대역폭·지연·Duplex

Transparent Bridging → 스위치와 투명 브리징

Triggered Update → Count-to-Infinity · Split Horizon · Route Poisoning

Troubleshooting → 계층별 트러블슈팅

Trust Boundary → 자산·위협·취약점·위험

Trusted Port → DHCP Snooping

TSO → Offloading · Checksum 미완성·Promiscuous Mode

TTL → ICMP · Ping · Traceroute · TTL

tuple → Python 자료형과 가변성

TXT → DNS Record · TTL · 정방향/역방향 조회

Type Enforcement → SELinux

Type-Length-Value → TLV

UAA → MAC 주소

UDLD → STP 보호 기능

UDP → UDP · QUIC · SCTP

UDP · QUIC · SCTP

UDP Scan → Nmap · Port Scan · Service Detection

UEFI → 서버 부팅·UEFI · MBR · GPT · Partition

UFW → iptables · nftables · firewalld · UFW

umask → Linux 권한·chmod · umask · Sticky bit

Unicast → 유니캐스트·브로드캐스트·멀티캐스트·애니캐스트

Unicode → 문자 인코딩과 ASCII

unit → systemd · 서비스·daemon

Unix → Linux 커널·셸·터미널

Unknown Unicast Flooding → Flooding과 MAC 주소 표 에이징

unreachable code → 조건문·반복문·종료 조건

Untrusted Port → DHCP Snooping

Uplink → Access port와 Trunk port

upstream → 패키지 관리자·Repository · upstream/downstream

URL Encoding → HTML Context · iframe · URL Encoding · CSP

User Datagram Protocol → UDP · QUIC · SCTP

user@host → DB 계정·Host · Bind Address

UTF-8 → 문자 인코딩과 ASCII

UTM → Inline/Non-inline · Bridge/Router Mode · UTM/NGFW

UTP → UTP · 광섬유·배선 규격

UTP · 광섬유·배선 규격

UUID → Mount · fstab · LVM · Quota

variable → Python 이름·객체·타입

VDI → On-premises · Cloud · IaC · Container

VDI → Mainframe · Workstation · VDI

venv → 실행 인자와 Python 가상 환경

Vertical Privilege Escalation → Broken Access Control · IDOR/BOLA

VG → Mount · fstab · LVM · Quota

vi → vi · Vim · nano와 Ctrl+C

VID → IEEE 802.1Q (dot1q)

vim → vi · Vim · nano와 Ctrl+C

Violation Mode → Port Security violation mode

Virtual Desktop Infrastructure → Mainframe · Workstation · VDI

virtual environment → 실행 인자와 Python 가상 환경

Virtual Hosting · SNI · Sandboxing

Virtualization → 가상화·Hypervisor · VM

Virtual LAN → VLAN

Virtual Memory → 가상 메모리와 Swap

virtual switch → VM 네트워크의 NAT · Bridge · Host-only

Virtual Teletype → Paramiko · Netmiko · Console · VTY

vi · Vim · nano와 Ctrl+C

VLAN

VLAN 1 → Native VLAN

VLAN tagging → IEEE 802.1Q (dot1q)

VLAN Trunking Protocol → VTP

VLSM → Subnet Mask · CIDR · VLSM · Summary

VMware ESXi → 가상화·Hypervisor · VM

VM 네트워크의 NAT · Bridge · Host-only

vNIC → VM 네트워크의 NAT · Bridge · Host-only

VPCS → GNS3 · 에뮬레이션·Cisco IOS CLI

vsftpd → FTP · SFTP · SMB/CIFS · Samba · NFS

VTP

VTP Client → VTP

VTP Pruning → VTP

VTP Server → VTP

VTP Transparent → VTP

Vulnerability → 자산·위협·취약점·위험

Vulnerability Identifier → CVE · CWE · CVSS

WAS → LAMP · Web Server · WAS

WAS Tuning → CGI · FastCGI · PHP-FPM · WSGI · Worker

Wazuh → SIEM · ELK · Wazuh

Weakness → CVE · CWE · CVSS

Web Application Server → LAMP · Web Server · WAS

WebGoat → Kali Linux · Metasploitable · DVWA · WebGoat · bee-box

Well-known Port → EtherType · IP Protocol Number · Port Number

well-known port → Port · Socket · 5-tuple

well-known port → 주요 응용 프로토콜과 Well-known Port

wget → 패키지 관리자·Repository · upstream/downstream

wget → cat · head · tail · grep · sed · tee · wget · curl · 파이프

while → 조건문·반복문·종료 조건

WHOIS → OSINT · WHOIS · Shodan · Maltego

Wide Area Network → PAN · LAN · CAN · MAN · WAN

Wildcard Mask → ACL · Permit/Deny · Wildcard Mask · Inbound/Outbound

Wireshark → Wireshark · Npcap · tcpdump · PCAP 분석

Wireshark · Npcap · tcpdump · PCAP 분석

with → 파일 입출력·Context Manager · Race Condition

WordPress → CMS와 DB 운영

Worker → CGI · FastCGI · PHP-FPM · WSGI · Worker

Workstation → Mainframe · Workstation · VDI

WPScan → Burp Suite · sqlmap · Hydra · John · Hashcat

write → 파일 입출력·Context Manager · Race Condition

WSGI → CGI · FastCGI · PHP-FPM · WSGI · Worker

XDR → EDR · XDR · SOAR · 능동 대응

Xen → 가상화·Hypervisor · VM

xferlog → Log · rsyslog · Severity · Trap

XFS → Filesystem · Inode · Journaling · CoW

X-Header → HTTP · Method · Status · Header

XSS

Zeek → Snort · Suricata · Zeek

Zenmap → Nmap · Port Scan · Service Detection

Zero-click → Cookie · Session · Session Hijacking

Zero Trust → Zero Trust · Microsegmentation

Zero Trust · Microsegmentation

Zone → DNS · Domain · FQDN · Zone

Zone → iptables · nftables · firewalld · UFW

가

가상 LAN → VLAN

가상 메모리와 Swap

가상화·Hypervisor·VM

계

계층별 트러블슈팅

네

네이티브 VLAN → Native VLAN

네트워크 범위 → PAN · LAN · CAN · MAN · WAN

네트워크 인터페이스 → NIC

네트워크 토폴로지

논

논리 토폴로지 → 네트워크 토폴로지

대

대역폭·지연·Duplex

대칭키·비대칭키·Hash · Digital Signature

동

동적 ARP 검사 → Dynamic ARP Inspection

랜

랜카드 → NIC

루

루트 브리지와 Bridge ID

루핑 → L2 루프와 루핑

리

리피터·허브·브리지·스위치·라우터·게이트웨이

메

메모리와 저장장치

모

모의해킹·Rules of Engagement

문

문자 인코딩과 ASCII

물

물리 주소 → MAC 주소

물리 토폴로지 → 네트워크 토폴로지

변

변수 Scope · LEGB · Closure

보

보안 기준과 실제 통제 검증

브

브로드캐스트 스톰

브리지 → 스위치와 투명 브리징

비

비트·바이트·옥텟과 진법

사

사설 주소·공인 주소·IANA · ISP

서

서버 부팅·UEFI · MBR · GPT · Partition

셸

셸 파이프·리다이렉션·종료 상태

스

스위치와 투명 브리징

스패닝 트리 → STP

실

실행 인자와 Python 가상 환경

액

액세스 포트 → Access port와 Trunk port

오

오류 제어·ARQ · Hamming Code

유

유니캐스트·브로드캐스트·멀티캐스트·애니캐스트

이

이더넷 → Ethernet

자

자산·위협·취약점·위험

조

조건문·반복문·종료 조건

주

주요 응용 프로토콜과 Well-known Port

주요정보통신기반시설 기술적 취약점 분석·평가방법 상세가이드 → OWASP · KISA · 주요정보통신기반시설 가이드

주정통 → OWASP · KISA · 주요정보통신기반시설 가이드

충

충돌 도메인과 브로드캐스트 도메인

캡

캡슐화와 역캡슐화

클

클래스·객체·인스턴스·getter/setter

태

태깅 → IEEE 802.1Q (dot1q)

통

통합 보안·네트워크 Lab

트

트렁크 포트 → Access port와 Trunk port

파

파일 링크 → Hard Link와 Symbolic Link

파일 입출력·Context Manager · Race Condition

패

패키지 관리자·Repository · upstream/downstream

포

포인터·참조·인자 전달

포트 시큐리티 → Port Security

프

프로그램·프로세스·PID

프로세스 메모리 구조

플

플러딩 → Flooding과 MAC 주소 표 에이징

함

함수·메서드·인자·1급 객체

기타

&& → 셸 파이프·리다이렉션·종료 상태

&& → cat · head · tail · grep · sed · tee · wget · curl · 파이프

@ → DNS Record · TTL · 정방향/역방향 조회

| → 셸 파이프·리다이렉션·종료 상태

| → cat · head · tail · grep · sed · tee · wget · curl · 파이프

.bashrc → bashrc · zshrc · 숨김 파일·명령 history

/dev → Linux 파일시스템 계층과 “모든 것은 파일”

__enter__ → 파일 입출력·Context Manager · Race Condition

/etc → Linux 파일시스템 계층과 “모든 것은 파일”

/etc/fstab → Mount · fstab · LVM · Quota

/etc/passwd → sudo · Linux 계정·Service User

/etc/shadow → sudo · Linux 계정·Service User

__exit__ → 파일 입출력·Context Manager · Race Condition

/proc → Linux 파일시스템 계층과 “모든 것은 파일”

/sbin/nologin → sudo · Linux 계정·Service User

/sys → Linux 파일시스템 계층과 “모든 것은 파일”

/var → Linux 파일시스템 계층과 “모든 것은 파일”

/var/log → Log · rsyslog · Severity · Trap

.zshrc → bashrc · zshrc · 숨김 파일·명령 history