

수업 내용 정리 보고서

작성일: 2026년 7월 24일

챕터 구성

- EIGRP 라우팅 프로토콜의 동작 원리와 수렴 메커니즘
- Firewalld와 UFW를 활용한 리눅스 방화벽 관리 체계
- 고급 방화벽 정책 수립을 위한 Rich Rules 활용과 실무 트러블슈팅

EIGRP 라우팅 프로토콜의 동작 원리와 수렴 메커니즘

EIGRP(Enhanced Interior Gateway Protocol)는 고전적인 거리 벡터(Distance Vector) 프로토콜의 장점과 링크 상태(Link State) 프로토콜의 빠른 수렴 능력을 결합한 하이브리드 성격의 라우팅 프로토콜입니다. 네트워크 설계 시 루프(Loop)를 방지하면서도 예측 가능한 경로 재계산 범위를 제공하는 것이 핵심입니다.

EIGRP의 아키텍처와 3대 테이블

EIGRP는 효율적인 라우팅 결정을 위해 세 가지 테이블을 독립적으로 운영합니다.

- 이웃 테이블 (Neighbor Table):** 직접 연결된 인접 EIGRP 라우터와의 관계를 유지합니다. 헬로(Hello) 패킷을 통해 인접성을 확인하며, `show ip eigrp neighbors` 명령으로 SRTT(평활 왕복 시간)와 RTO(재전송 타이머) 등을 검증할 수 있습니다.
- 토폴로지 테이블 (Topology Table):** 이웃으로부터 학습한 모든 경로와 DUAL 알고리즘에 의해 계산된 상태(Passive/Active)를 보관합니다. `show ip eigrp topology` 명령을 통해 성공자(Successor)와 가능한 성공자(Feasible Successor)를 확인합니다.
- 라우팅 테이블 (Routing Table):** 토폴로지 테이블 중 DUAL이 선정한 최적의 경로(Successor)만이 최종적으로 설치됩니다. `show ip route eigrp`로 확인 가능합니다.

DUAL 알고리즘과 수렴 메커니즘

DUAL(Diffusing Update Algorithm)은 루프 없는 경로를 계산하는 EIGRP의 핵심 엔진입니다. 경로 수렴은 다음과 같은 논리로 진행됩니다.

- 성공자 (Successor):** 최적의 메트릭을 가진 경로이며, 라우팅 테이블에 즉시 등록됩니다.
- 가능한 성공자 (Feasible Successor, FS):** 장애 발생 시 즉시 대체할 수 있는 백업 경로입니다. **타당성 조건(Feasibility Condition)**을 만족해야 하는데, 이는 '이웃이 광고한 거리(RD)가 현재 성공자의 총 메트릭(FD)보다 작아야 한다'는 규칙입니다.
- 수렴 과정:** 성공자 경로에 장애가 발생했을 때, FS가 존재하면 즉시 해당 경로를 성공자로 승격시켜 패킷 손실을 최소화합니다. FS가 없을 경우에만 해당 경로를 **Active** 상태로 전환하고, 이웃 라우터들에게 **Query**를 보내 대체 경로를 찾는 확산 계산을 수행합니다.

5가지 패킷 타입과 신뢰성 전송 (RTP)

EIGRP는 신뢰성 있는 전송을 위해 RTP(Reliable Transport Protocol)를 사용합니다. 데이터의 전달과 수신 확인을 관리하여 신뢰성을 확보합니다.

패킷 종류	기능	전송 방식
Hello	이웃 발견 및 인접 관계 유지	멀티캐스트(224.0.0.10)
Update	경로 정보 전송 및 동기화	신뢰성 전송
ACK	수신 확인	유니캐스트(데이터 없음)
Query	대체 경로 질의	신뢰성 전송
Reply	Query에 대한 응답	유니캐스트

SIA(Stuck In Active) 상태와 장애 예방

라우터가 Query를 보낸 뒤 일정 시간 내에 Reply를 받지 못하면 해당 목적지는 **SIA(Stuck In Active)** 상태가 됩니다. 이는 심각한 네트워크 단절이나 CPU 과부하, 비효율적인 설계의 징후입니다.

- SIA 발생 원인:** 응답하지 않는 이웃 라우터, 과도하게 넓은 Query 확산 범위, 링크의 대역폭 제한 등.
- 예방 전략:** SIA 상태가 되면 EIGRP는 해당 이웃과의 인접 관계를 강제로 초기화(Reset)합니다. 이를 방지하려면 수동 요약(Manual Summary)을 통해 Query 범위를 국소화하거나, EIGRP Stub 설정을 통해 말단 라우터의 Query 확산을 차단해야 합니다.

운영 및 트러블슈팅 체크리스트

EIGRP를 성공적으로 운영하기 위한 점검 항목은 다음과 같습니다.

- 인접성 확인:** AS 번호, K 값, MD5 인증 키가 양단 라우터에서 일치하는지 확인합니다.
- 토폴로지 분석:** `show ip eigrp topology active` 명령을 통해 현재 수렴이 지연 중인 경로와 대기 중인 응답(Reply)을 추적합니다.
- 필터링 검증:** `distribute-list`나 `prefix-list`가 의도치 않게 중요한 경로를 차단하고 있지는 않은지 확인합니다.
- 대역폭 제어:** `ip bandwidth-percent eigrp` 설정을 통해 EIGRP 제어 패킷이 인터페이스 대역폭의 너무 큰 비율을 점유하지 않도록 관리합니다.

주의사항: 네트워크 장애 상황에서 `timers active-time`을 단순히 늘리는 것은 근본적인 해결책이 아닙니다. SIA 문제는 주로 네트워크 설계(요약)의 미비나 링크 품질 문제에서 기인하므로, DUAL이 계산하는 확산 범위를 줄이는 데 집중해야 합니다.

Firewalld와 UFW를 활용한 리눅스 방화벽 관리 체계

리눅스 시스템 보안의 핵심은 불필요한 네트워크 접근을 차단하고 허용된 서비스만을 안전하게 공개하는 것입니다. 이를 위해 리눅스 커널은 Netfilter라는 강력한

패킷 처리 엔진을 제공하며, 관리자는 방화벽 관리 서비스(Firewalld, UFW)를 통해 이 엔진을 보다 쉽고 직관적으로 제어합니다. 본 챕터에서는 RHEL 계열에서 표준으로 사용하는 Firewalld와 Ubuntu 계열의 UFW를 중심으로 리눅스 방화벽 관리 체계를 살펴봅니다.

1. 리눅스 방화벽의 계층 구조

방화벽 관리 서비스와 커널 엔진의 관계를 이해하는 것은 트러블슈팅의 첫걸음입니다. 모든 제어는 다음 세 단계를 거쳐 수행됩니다.

- **Netfilter:** 리눅스 커널 내부에 존재하는 실제 패킷 필터링 엔진입니다.
- **nftables/iptables:** 커널의 Netfilter 기능을 조작하기 위한 직접적인 도구입니다.
- **Firewalld/UFW:** 사용자가 복잡한 커널 규칙을 직접 다루지 않도록 추상화된 인터페이스를 제공하는 관리 서비스입니다.

이 서비스들은 사용자가 입력한 규칙을 내부적으로 nftables나 iptables 문법으로 변환하여 커널에 적용합니다. 따라서 방화벽 서비스를 재시작하거나 규칙을 추가하더라도 기존에 확립된 네트워크 세션이 끊기지 않는 동적 반영(Dynamic Update)이 가능합니다.

2. Firewalld: Zone 기반의 유연한 보안 관리

Firewalld는 RHEL(Rocky, CentOS, Fedora) 기반 시스템에서 기본으로 제공되며, 신뢰 수준에 따른 **Zone(영역)** 개념을 도입하여 정책을 관리합니다.

Zone의 개념과 활용

네트워크 인터페이스(NIC)나 연결 상태에 따라 각각 다른 보안 정책을 적용하는 논리적 그룹입니다. 예를 들어, 서버의 ens160 인터페이스는 인터넷을 향하는 public zone으로, ens192는 관리자가 접근하는 trusted zone으로 분리하여 관리할 수 있습니다.

Zone	신뢰 수준	정책 특징
public	낮음	필요한 서비스만 명시적으로 허용
trusted	매우 높음	대부분의 연결 허용 (관리망)
drop/block	매우 낮음	모든 연결 차단 (공격자 차단 시 사용)

Rich Rules: 세밀한 접근 제어

기본 Zone 정책만으로 해결하기 어려운 정교한 제어가 필요할 때 Rich Rules(확장 규칙)를 사용합니다. 단순히 서비스 이름을 허용하는 것을 넘어, 출발지 IP 주소, 로그 기록, ICMP 차단 등을 조합할 수 있습니다.

예를 들어, 특정 IP(192.168.1.100)로부터 들어오는 SSH 연결만을 허용하고 그 과정을 로그에 남기는 규칙은 다음과 같이 작성합니다.

```
firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="192.168.1.100" service name="ssh" log prefix="SSH_ACCESS" accept'
```

이러한 규칙은 반드시 적용 후 --reload 명령어를 통해 정책을 반영해야 합니다. Runtime 모드(재부팅 시 초기화)와 Permanent 모드(영구 적용)의 구분을 명확히 하는 것이 운영상의 핵심입니다.

3. UFW: 초보자를 위한 직관적인 보안 도구

Ubuntu 등 Debian 계열에서 사용하는 UFW(Uncomplicated Firewall)는 이름 그대로 '복잡하지 않은 방화벽'을 지향합니다. iptables의 방대하고 복잡한 문법을 극도로 단순화하여, 명령어 한 줄로 보안 정책을 구현합니다.

핵심 관리 방법

- **기본 정책 설정:** 화이트리스트 방식을 권장합니다. 모든 수신 연결을 차단하고 필요한 포트만 개별적으로 개방합니다.
- **서비스 이름 및 포트:** 'allow 80/tcp'처럼 포트 번호뿐만 아니라 'allow Nginx'와 같은 애플리케이션 프로파일을 사용할 수 있어 관리가 편리합니다.
- **상태 확인:** 'ufw status numbered' 명령을 통해 현재 적용된 규칙에 번호를 매겨 출력하면, 이후 'ufw delete [번호]' 명령어로 오타 없이 규칙을 삭제할 수 있습니다.

4. 실무 운영 및 장애 대응 체크리스트

방화벽 운영 시 흔히 발생하는 문제와 점검 항목입니다.

- **로그 확인:** 접속이 거부될 때 /var/log/ufw.log 혹은 시스템 저널 로그를 확인하여 차단된 패킷의 출발지 IP와 목적지 포트를 확인합니다.
- **영구 설정 vs 임시 설정:** 설정 후 재부팅 시 규칙이 사라진다면 Permanent 옵션이나 설정 파일 저장 여부를 확인해야 합니다.
- **인터페이스 할당 확인:** 특정 Zone에 인터페이스가 올바르게 할당되었는지 확인합니다. 할당되지 않은 인터페이스는 default zone의 정책을 따르게 되어 의도치 않은 차단이 발생할 수 있습니다.
- **순서 문제:** 방화벽 규칙은 순차적으로 처리됩니다. 특히 특정 IP를 차단한 후 동일한 IP를 허용하는 규칙을 추가하면, 앞선 차단 규칙이 먼저 적용되므로 주의해야 합니다.

결론적으로, Firewalld는 대규모 엔터프라이즈 서버 환경에서 Zone 기반의 체계적인 관리에 적합하며, UFW는 신속하고 직관적인 보안 설정이 필요한 중소 규모 시스템이나 초기 학습에 최적화되어 있습니다. 어느 도구를 사용하든 방화벽의 기본 원리는 '최소 권한의 원칙'에 따라 반드시 필요한 포트와 IP만 허용하는 것입니다.

다.

고급 방화벽 정책 수립을 위한 Rich Rules 활용과 실무 트러블슈팅

방화벽 운영 시 단순한 포트 허용이나 서비스 단위 개방만으로는 정교한 보안 정책을 구현하기 어려운 경우가 많습니다. RHEL 및 Rocky Linux 계열에서 제공하는 FirewallD의 Rich Rules(고급 규칙)은 특정 IP 주소, 시간, 로그 기록, 속도 제한 등 세밀한 조건 조합을 가능하게 합니다. 본 챗터에서는 Rich Rules의 개념과 실무 적용 사례, 그리고 트러블슈팅 방법을 다룹니다.

Rich Rules의 개념과 필요성

기본 FirewallD 설정은 'Zone(영역)' 기반으로 동작하며, 포트나 서비스 단위의 허용/차단에 특화되어 있습니다. 반면, Rich Rules는 기존의 제한적인 설정을 넘어선 확장 규칙 언어로, 다음과 같은 요구사항이 있을 때 필수적으로 사용됩니다.

- 특정 출발지 IP 주소에서 들어오는 특정 서비스만 허용하고자 할 때
- 방화벽에 의해 차단되거나 허용된 패킷의 로그를 남겨 분석하고자 할 때
- 특정 프로토콜(ICMP 등)에 대해 엄격한 차단 정책을 적용하고자 할 때

Rich Rule은 반드시 특정 Zone 내에 소속되어 동작합니다. 즉, 패킷이 먼저 Zone으로 분류된 후, 해당 Zone에 설정된 Rich Rule이 검사되는 구조입니다. 이는 iptables의 복잡한 문법을 대신하여 사람이 읽고 관리하기 쉬운 구조로 정책을 표현하는 데 주안점을 둡니다.

실무 적용을 위한 주요 규칙 예시

Rich Rules는 `firewall-cmd --add-rich-rule` 명령을 통해 적용합니다. 실무에서 빈번하게 사용되는 패턴은 다음과 같습니다.

1. **특정 IP의 SSH 접속 허용:** 특정 관리자 PC(예: 192.168.1.100)만 SSH 접근이 가능하도록 설정하여 보안을 강화합니다.

```
firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="192.168.1.100" port port="22" protocol="tcp" accept'
```

2. **접근 차단 및 로그 기록:** 승인되지 않은 IP를 차단하고 해당 시도를 로그에 기록하여 추후 침입 탐지 분석 자료로 활용합니다.

```
firewall-cmd --permanent --add-rich-rule='rule service name="ssh" log prefix="SSH_ACCESS " accept'
```

3. **ICMP 차단:** Ping 테스트(ICMP)를 차단하여 네트워크 스캐닝을 어렵게 만듭니다.

```
firewall-cmd --permanent --add-rich-rule='rule protocol value="icmp" drop'
```

실무 트러블슈팅 및 점검 순서

Rich Rules 적용 후 의도대로 동작하지 않는다면 다음 순서에 따라 문제를 점검해야 합니다.

1. **Zone 확인:** 규칙이 올바른 Zone에 적용되었는지 확인합니다. `firewall-cmd --list-all --zone=[Zone명]`을 통해 설정된 Rich Rules 목록을 조회합니다.
2. **규칙 순서 및 우선순위:** FirewallD는 규칙을 순차적으로 평가합니다. 상위 정책(예: 다른 Zone의 설정)에 의해 차단되고 있지는 않은지 확인합니다.
3. **로그 분석:** 정책에 log 옵션을 포함했다면, `/var/log/messages` 또는 `journalctl -u firewalld`를 통해 패킷이 실제로 허용/차단되는지 확인합니다.
4. **Runtime vs Permanent 적용:** `--permanent` 옵션을 사용했다면 반드시 `firewall-cmd --reload`를 실행하여 설정을 동적으로 반영해야 합니다.

주의사항 및 실습 요약

Rich Rules는 iptables의 복잡한 연산 기능을 대체하지만, 과도하게 복잡한 규칙은 CPU 자원을 소모하고 관리 효율성을 떨어뜨릴 수 있습니다. 정책 수립 시 다음 원칙을 준수하십시오.

- **최소 권한의 원칙:** 필요한 IP와 서비스만 명시적으로 허용(accept)하고, 나머지는 거부(drop/reject)하는 정책을 기본으로 합니다.
- **테스트 환경 확인:** drop 정책이나 전체 차단 정책은 관리자 접속을 단절시킬 위험이 있으므로, 적용 전 테스트 환경에서 반드시 검증하십시오.
- **비교 학습:** UFW는 더 단순한 명령어를 제공하지만 세밀한 설정에 한계가 있고, iptables는 강력하지만 문법이 매우 복잡합니다. Rich Rules는 이 중간 지점에서 읽기 쉬운 문법으로 세밀한 제어를 가능하게 한다는 점을 기억하십시오.

최종적으로 방화벽 정책은 `firewall-cmd --list-all` 명령을 통해 전체적인 Zone별 정책을 정기적으로 점검하는 습관이 중요합니다. 오해하기 쉬운 부분은 Rich Rule이 전역 규칙이 아니라 특정 Zone에 종속된다는 점입니다. 패킷의 유입 경로(인터페이스/소스 IP)가 올바른 Zone에 매핑되어 있는지 선제적으로 점검하십시오.