01 용어 정리

용어 정리

Performance Tuning은 게이트웨이가 트래픽을 더 빠르게, 더 많이 처리하도록 만드는 기술들을 다룹니다. 이 가이드를 읽는 데 바탕이 되는 핵심 용어를 흐름에 따라 풀어 둡니다.

두 가지 가속의 축 — SecureXL·CoreXL

SecureXL 은 게이트웨이를 지나는 트래픽을 가속 하는 기술입니다. 핵심은 **Connection Template** 으로, 같은 출발지·목적지·포트의 새 연결을 빠르게 수립 하도록 활성 연결에서 템플릿을 만듭니다. SecureXL은 R82에서 커널 모드(KPPAK, Kernel Mode SecureXL Packet flow) 로 동작합니다(SecureXL 개념).

CoreXL 은 멀티코어 성능을 끌어내는 기술 로, 방화벽 커널을 여러 벌 복제해 각 CPU 코어에서 하나씩 돌립니다. 복제된 검사 커널 하나가 **CoreXL Firewall Instance(FWK)** 이고, 들어오는 트래픽을 받아 인스턴스로 분배하는 것이 **CoreXL SND(Secure Network Distributor)** 입니다(CoreXL 개념).

분배·할당 용어

Affinity(친화성) 는 인터페이스·프로세스를 특정 CPU 코어에 묶는 설정입니다. SND·인스턴스가 어느 코어에서 돌지를 정하며, `fwaffinity.conf` 로 구성합니다(CoreXL Affinity). **Multi-Queue** 는 한 네트워크 인터페이스에 여러 트래픽 큐를 둬 여러 코어를 가속에 동원 하는 기술입니다 — 기본적으로 인터페이스 하나는 큐 하나를 한 코어가 처리하는데, 이를 넘어서게 합니다(Multi-Queue).

HyperFlow 는 elephant flow(대용량 연속 연결)를 여러 코어로 병렬 검사 하는 R81.20+ 기술입니다 — HyperFlow가 없으면 한 elephant 연결을 코어 하나로만 검사합니다(HyperFlow).

보호·모니터링 용어

Rate Limiting 은 SecureXL이 대역폭·연결 수·연결률을 제한해 DoS 공격을 완화 하는 기능이고, **Accelerated SYN Defender** 는 SYN flood를 가속 경로에서 방어 합니다(DoS 완화). **CPView** 는 CPU·메모리·디스크와 블레이드별 통계를 실시간으로 보여 주는 텍스트 유틸리티 이고, **CPU Spike Detective** 는 CPU 급증(spike)을 탐지 합니다(모니터링).

명령·진단 용어

성능 진단의 핵심 명령은 `fwaccel` (SecureXL 제어·통계, 예: `fwaccel stats`)와 `fw ctl affinity` (Affinity), `fw ctl multik` (CoreXL 인스턴스) 입니다(명령줄·커널 참조). 고급 동작은 **Kernel Parameter**(`fw ctl set` · `fwkern.conf`)로 조정하고, 깊은 진단은 **Kernel Debug**(`fw ctl debug`)로 합니다.

02 성능 튜닝 소개

성능 튜닝 소개

방화벽은 더 많은 트래픽을 더 빠르게 처리할수록 좋습니다. Check Point은 이를 위해 여러 가속·확장 기술을 제공하는데, 이 가이드는 그 기술들을 무엇이고 언제·어떻게 켜는지 를 다룹니다. Security Gateway 가이드에서 짧게 본 가속 기술의 본체입니다.

성능을 끌어올리는 기술들

성능 튜닝의 두 축은 **SecureXL** 과 **CoreXL** 입니다.

SecureXL 은 게이트웨이를 지나는 트래픽을 가속 합니다 — 한 번 검사한 연결의 패턴을 **Connection Template** 으로 만들어, 같은 부류의 새 연결을 빠른 경로로 처리합니다(SecureXL 개념). **CoreXL** 은 방화벽 커널을 여러 벌 복제해 여러 CPU 코어에서 병렬 로 트래픽을 검사합니다(CoreXL 개념). 둘은 함께 동작합니다 — SND가 트래픽을 받아 가속(SecureXL)하거나 여러 방화벽 인스턴스(CoreXL)로 분배합니다.

여기에 세 기술이 더해집니다. **Multi-Queue** 는 한 인터페이스에 여러 큐를 둬 더 많은 코어를 가속에 동원 하고(Multi-Queue), **HyperFlow** 는 대용량 연속 연결(elephant flow)을 여러 코어로 병렬 검사 합니다(HyperFlow). 상태를 들여다보는 **CPView·CPU Spike Detective** 로 성능을 모니터링합니다(모니터링).

이 가이드의 흐름

큰 줄기는 SecureXL(개념·구성·DoS 완화·명령) → CoreXL(개념·구성·Affinity·튜닝) → Multi-Queue → 모니터링 → HyperFlow → 커널 참조 입니다. 대부분 깊은 튜닝이라, 각 장은 기능이 무엇이고 언제 켜며 어떤 명령으로 다루는지 를 개념 위주로 잡고, 방대한 명령·파라미터 사전은 명령줄·커널 참조와 원문 해당 절로 넘깁니다.

> 참 고
>
> 성능 튜닝은 환경마다 최적값이 다르므로, 무작정 바꾸기보다 CPView로 병목을 확인하고, Check Point Support의 SK 문서·Best Practice를 따라 조정하는 것이 안전합니다.

03 SecureXL — 개념과 동작

SecureXL — 개념과 동작

SecureXL 은 게이트웨이를 지나는 트래픽을 가속해 처리량을 높이는 기술입니다. 이 장은 SecureXL이 무엇을 가속하고 어떤 원리로 빨라지는지를 정리합니다.

무엇을 가속하나

SecureXL은 폭넓은 기능을 가속합니다 — Access Control, 암호화, NAT, 정책 설치, 어카운팅·로깅, 연결/세션률, 일반 보안 검사, ClusterXL HA·Load Sharing, TCP 시퀀스 검증, 동적 VPN, passive/active streaming 등입니다. Software Blade 중에서도 Firewall·IPS·Application Control·URL Filtering·Anti-Virus·Anti-Bot·Identity Awareness·Site-to-Site VPN·HTTPS Inspection·QoS 의 트래픽이 가속됩니다(단 Identity Agent 트래픽은 Connection Template을 만들지 않음).

Connection Template — 가속의 핵심

빨라지는 비결이 **Connection Template** 입니다. 같은 출발지 IP·목적지 IP·목적지 포트로 향하는 새 연결을 빠르게 수립 하도록, 활성 연결에서 Rule Base에 따라 템플릿을 만들 어 둡니다. 그러면 같은 부류의 다음 연결은 전체 검사를 다시 거치지 않고 템플릿으로 빠르게 처리됩니다(템플릿 제약은 sk32578).

KPPAK — 커널 모드 패킷 흐름

R82의 SecureXL은 커널 모드(KPPAK, Kernel Mode SecureXL Packet flow)로 동작합니다.

!SecureXL 커널 모드(KPPAK) 패킷 흐름 *NVIDIA ConnectX 100G 카드가 없는 Security Appliance에서 SecureXL이 커널 모드로 동작할 때의 일반적 패킷 흐름*

정책 설치·확장성

SecureXL은 정책 설치 중에도 계속 켜진 채 동작 해(Policy Installation Acceleration), 설치 시 CPU 부하를 줄입니다. 또 R80.20+부터 높은 세션률에서의 확장성이 개선 되어, CoreXL SND 코어 수에 대한 제한이 사라 졌습니다(CoreXL 개념).

정리하면, SecureXL은 Connection Template으로 같은 부류 연결을 빠른 경로로 처리하고, 커널 모드(KPPAK)로 동작 하며 CoreXL과 함께 게이트웨이 성능을 끌어올립니다. 실제 켜고 분석하는 방법은 SecureXL 구성에서 이어집니다.

04 SecureXL — 구성과 분석

SecureXL — 구성과 분석

SecureXL의 개념을 잡았으니, 이제 켜고 끄는 방법과 가속이 실제로 얼마나 일어나는지 분석 하는 방법을 정리합니다.

SecureXL 구성

SecureXL은 기본으로 켜져 있으며, 게이트웨이 명령줄에서 제어합니다. 핵심 명령이 `fwaccel` (IPv4)·`fwaccel6` (IPv6)입니다 — `fwaccel on` / `fwaccel off` 로 켜고 끄 며 (클러스터에서는 모든 멤버를 똑같이), 상태는 `fwaccel stat` 으로 봅니다.

> 참 고
>
> SecureXL을 끄면 CoreXL·Multi-Queue 등 이를 전제로 하는 기능도 영향을 받습니다. 또 클러스터에서는 모든 멤버의 SecureXL 상태가 같아야 합니다(ClusterXL 가이드의 요구사항).

구성은 대부분 명령과 Kernel Parameter 로 이뤄지며, 환경에 맞춰 Connection Template 동작·가속 범위를 조정합니다.

가속 트래픽 분석

튜닝의 출발점은 지금 트래픽이 얼마나 가속되는지 보는 것입니다. 핵심 명령이 `fwaccel stats` 로, SecureXL 통계(가속·비가속 트래픽, 템플릿, 드롭 등) 를 보여 줍니다. 드롭만 보려면 `fwaccel stats -d` , 자세한 내부 상태는 `/proc/ppk/` · `/proc/ppk6/` 항목을 봅니다.

이 통계로 가속 비율이 낮으면 왜 비가속 경로(slow path)로 가는지 를 파악해, Rule Base나 설정을 조정합니다. 예를 들어 가속을 막는 규칙·기능을 찾아 줄이면 가속 비율이 올라갑니다.

정리하면, SecureXL은 `fwaccel` 로 켜고 끄고, `fwaccel stats` 로 가속 현황을 분석 해 병목을 찾는 것이 운영의 기본입니다. DoS 방어용 설정은 DoS 완화, 전체 명령은 SecureXL 명령에서 다룹니다.

05 SecureXL — DoS 완화(Rate Limiting·SYN Defender)

SecureXL — DoS 완화(Rate Limiting·SYN Defender)

SecureXL은 가속만 하는 게 아니라 DoS(서비스 거부) 공격을 빠른 경로에서 막 기도 합니다. 두 기능 — Rate Limiting과 Accelerated SYN Defender — 을 정리합니다.

Rate Limiting for DoS Mitigation

Rate Limiting 은 DoS 공격에 대한 방어 입니다. 지정한 출발지에서 오거나 지정한 목적지·서비스로 가는 트래픽을 제한 하는 규칙으로, SecureXL이 직접 집행 합니다. 제한 대상은 대역폭·패킷률, 동시 연결 수, 연결률 입니다.

설정은 두 갈래 명령으로 합니다 — `fw sam_policy` (`quota` 인자 사용) 와 `fwaccel dos config` 입니다(상세는 sk112454). 다만 특정 URL에는 적용할 수 없 고 모든 트래픽에 적용됩니다.

DoS 관련 상태는 `fwaccel stats` (전체 통계), `fwaccel stats -d` (드롭 통계), `fw sam_policy get` (활성 정책 규칙) 으로 모니터링합니다. 즉 공격이 의심되면 드롭 통계와 SAM 정책을 확인 해 Rate Limiting이 제대로 막고 있는지 봅니다.

Accelerated SYN Defender

Accelerated SYN Defender 는 SYN flood 공격을 가속 경로에서 방어 합니다. SYN flood는 TCP 연결의 SYN 패킷만 잔뜩 보내 연결 테이블을 고갈 시키는 공격인데, SYN Defender가 SecureXL 수준에서 이를 걸러 게이트웨이를 보호합니다.

정리하면, SecureXL의 DoS 완화는 Rate Limiting으로 과도한 트래픽(대역폭·연결 수·연결률)을 제한하고, Accelerated SYN Defender로 SYN flood를 가속 경로에서 막 는 두 축입니다. 둘 다 빠른 경로(SecureXL)에서 동작해 공격 트래픽이 느린 경로의 자원을 잡아먹기 전에 차단합니다. 세부 명령·옵션은 SecureXL 명령과 원문 해당 절을 참고하세요.

06 SecureXL — 명령과 디버그

SecureXL — 명령과 디버그

SecureXL을 제어·진단·디버그하는 명령 을 정리합니다. 원문에서 가장 방대한 부분이라, 여기서는 핵심 명령과 무엇에 쓰는지 를 짚고 전체 옵션 사전은 원문 해당 절로 넘깁니다.

핵심 명령 — fwaccel

SecureXL의 중심 명령이 `fwaccel` (IPv4)· `fwaccel6` (IPv6)입니다. 자주 쓰는 것만 추리면 — `fwaccel on` / `off` (가속 켜고 끄기), `fwaccel stat` (상태), `fwaccel stats` (통계), `fwaccel conns` (가속 연결 목록), `fwaccel templates` (Connection Template 목록), `fwaccel dos` (DoS 완화 설정) 입니다(구성·분석·DoS 완화에서 본 명령들).

내부 상태는 `/proc/ppk/` · `/proc/ppk6/` 항목 으로도 볼 수 있어, 통계·드롭 원인을 깊이 들여다볼 수 있습니다.

디버그

SecureXL이 예상대로 가속하지 않거나 트래픽이 느린 경로로 빠질 때는 SecureXL 디버그 로 원인을 찾습니다. `fwaccel dbg` 로 디버그 플래그를 켜 패킷이 왜 가속/비가속 경로로 가는지를 추적합니다.

> 참 고
>
> 클러스터에서는 모든 멤버를 똑같이 설정하고, Scalable Platforms에서는 Expert 모드에서 해당 Security Group 위에서 실행합니다.

정리하면, SecureXL 운영의 손과 발이 `fwaccel` 계열 명령 입니다 — 켜고 끄고, 통계로 가속 현황을 보고, 디버그로 비가속 원인을 찾습니다. 각 명령의 전체 구문·옵션은 원문 SecureXL Commands and Debug 절과 R82 CLI Reference Guide를 참고하세요.

07 CoreXL — 개념과 구성

CoreXL — 개념과 구성

CoreXL 은 방화벽 커널을 여러 벌 복제해 여러 CPU 코어에서 병렬로 트래픽을 검사 하는 기술입니다. 이 장은 CoreXL의 구조와 기본 구성을 정리합니다.

CoreXL의 구조

CoreXL을 켜면 방화벽 커널이 여러 벌 복제 됩니다. 복제된 검사 커널 하나가 **CoreXL Firewall Instance(FWK)** 로, 각각 완전하고 독립된 검사 커널 이며 같은 인터페이스·같은 정책으로 동시에 트래픽을 처리합니다(Security Gateway 가이드에서 본 개념). 들어오는 트래픽을 받아 각 인스턴스로 분배하는 것이 **CoreXL SND(Secure Network Distributor)** 입니다. 코어 수에 거의 선형으로 성능이 확장 됩니다.

켜고 끄기, 기본 구성

CoreXL은 명령으로 켜고 끕니다. `cpconfig` 메뉴 또는 명령 으로 활성/비활성하며, 변경은 재부팅 후 적용됩니다.

기본 인스턴스 수는 CPU 코어 총 수에 따라 자동 으로 정해집니다 — 예를 들어 코어 2개면 FW 인스턴스 2·SND 2, 코어 4개면 FW 인스턴스 3·SND 1 식입니다(코어 1개면 CoreXL 비활성). SecureXL이 켜져 있으면 모든 인터페이스의 affinity가 자동 으로 잡혀, 트래픽이 SND를 도는 코어로 향합니다. (단 이 기본값은 Dynamic Balancing을 지원하지 않는 게이트웨이에만 적용됩니다.)

인스턴스 구성과 한계

IPv4·IPv6 CoreXL Firewall 인스턴스 수를 따로 구성 할 수 있어, 트래픽 특성에 맞춰 코어를 배분합니다. 다만 한계도 있습니다 — ClusterXL 가이드에서 봤듯 클러스터 멤버 간 인스턴스 수가 다르면 상태가 어긋나 거나(인스턴스가 더 많은 멤버는 DOWN), 페일오버 시 일부 연결이 끊길 수 있습니다.

정리하면, CoreXL은 SND가 받아 여러 FW 인스턴스로 분배해 코어를 병렬 활용 하며, 기본 인스턴스 수는 코어 수에 따라 자동으로 잡힙니다. 어느 코어에 무엇을 묶을지(Affinity)와 세밀한 튜닝은 CoreXL Affinity와 성능 튜닝에서 이어집니다.

08 CoreXL — Affinity와 성능 튜닝

CoreXL — Affinity와 성능 튜닝

CoreXL의 구조를 잡았으니, 이제 어느 작업을 어느 CPU 코어에 묶을지(Affinity)와 성능을 끌어올리는 튜닝 을 정리합니다.

Affinity — 코어에 묶기

Affinity(친화성) 는 인터페이스·프로세스를 특정 CPU 코어에 고정 하는 설정입니다. SND가 도는 코어와 FW 인스턴스가 도는 코어를 나눠 , 트래픽 수신·분배와 검사가 서로 다른 코어에서 효율적으로 돌게 합니다.

설정은 `$FWDIR/conf/fwaffinity.conf` 파일 로 합니다. 부팅 때 `$FWDIR/scripts/fwaffinity_apply` 스크립트가 이 파일대로 affinity를 적용 하며, 값을 바꾸면 재부팅하거나 그 스크립트를 수동 실행 해야 적용됩니다. 인터페이스 affinity는 SecureXL이 켜져 있으면 자동(automatic)이 기본이고, 필요하면 수동으로 특정 코어에 묶습니다. 명령으로는 `fw ctl affinity` 로 현재 affinity를 보고 설정 합니다.

성능 튜닝

Performance Tuning 절은 CoreXL을 환경에 맞게 최적화 하는 방법을 다룹니다. 핵심은 SND 코어와 FW 인스턴스 코어의 균형 입니다 — 트래픽 수신이 병목이면 SND 코어를, 검사가 병목이면 FW 인스턴스를 늘리는 식으로 CPView로 본 병목에 맞춰 배분합니다.

R82에는 **Dynamic Balancing of CoreXL Instances** 가 있어, 부하에 따라 SND·FW 인스턴스 코어 배분을 자동으로 조정 할 수 있습니다 — 사람이 일일이 맞추지 않아도 부하 변화에 맞춰 코어가 재배분됩니다.

명령

CoreXL의 핵심 명령은 `fw ctl multik` (인스턴스 상태·통계), `fw ctl affinity` (affinity 보기·설정), `cpconfig` (인스턴스 수 변경) 입니다(CoreXL 개념). 전체 명령·옵션은 원문 CoreXL Commands 절과 R82 CLI Reference Guide를 참고하세요.

정리하면, Affinity로 SND·FW 인스턴스를 코어에 적절히 나누고, Dynamic Balancing이나 수동 튜닝으로 병목에 맞춰 코어를 배분 하는 것이 CoreXL 성능 튜닝의 핵심입니다.

09 Multi-Queue

Multi-Queue

Multi-Queue 는 한 네트워크 인터페이스에 여러 트래픽 큐를 둬, 더 많은 CPU 코어를 가속에 동원 하는 기술입니다. CoreXL·SecureXL과 맞물려 성능을 끌어올립니다.

왜 필요한가

기본적으로 네트워크 인터페이스 하나는 트래픽 큐 하나를 한 CPU 코어가 처리 합니다. 그래서 가속에 쓸 수 있는 코어 수가 트래픽을 처리하는 인터페이스 수를 넘을 수 없 다는 한계가 생깁니다 — 인터페이스가 적은데 코어가 많으면, SND 코어가 놀게 됩니다. **Multi-Queue** 는 한 인터페이스에 큐를 여러 개 둬 여러 코어가 그 인터페이스의 트래픽을 나눠 받게 합니다.

켤지 결정하기

무작정 켜지 않고 이득이 있을지 먼저 판단 합니다. 핵심 점검은 네트워크 카드(드라이버)가 Multi-Queue를 지원하는지 입니다 — 특정 드라이버를 쓰는 카드만 지원하며, `ethtool -i <인터페이스>` 로 드라이버를 확인 합니다(요구사항·한계는 원문 참고, 드라이버 업그레이드 전 최신 버전이 지원하는지 확인). 보통 SND 코어가 100%에 가까운데 인터페이스가 적을 때 Multi-Queue가 효과적입니다.

구성과 특수 시나리오

기본 구성은 지원 인터페이스에 Multi-Queue를 켜고 큐 수(코어 수)를 정하는 것입니다 (SecureXL이 켜져 있어야 함). Bond 인터페이스나 특정 환경을 위한 **특수 시나리오** 구성도 있고, 문제가 생기면 **Troubleshooting** 절차로 큐 분배·코어 할당 을 점검합니다.

정리하면, Multi-Queue는 인터페이스 하나에 큐를 늘려 노는 코어까지 트래픽 수신·가속에 동원 하는 기술로, SND 코어가 병목인데 인터페이스가 적을 때 특히 효과적입니다. 세부 구성·드라이버 요구사항은 원문 해당 절을 참고하세요.

10 모니터링 — CPView·CPU Spike Detective

모니터링 — CPView·CPU Spike Detective

성능 튜닝의 출발점은 지금 무엇이 병목인지 보는 것입니다. 이 장은 상태를 들여다보는 두 도구 — CPView와 CPU Spike Detective — 를 정리합니다.

CPView

CPView 는 Check Point 장비에 내장된 텍스트 기반 유틸리티 입니다. CPU·메모리·디스크 같은 일반 시스템 정보와, 게이트웨이의 블레이드별 정보를 실시간으로 보여 줍니다(데이터가 계속 갱신됨). `cpview` 명령으로 실행하며, 화살표·단축키로 여러 뷰를 오갑니다.

성능 튜닝에서 CPView는 병목을 찾는 출발점 입니다 — 예를 들어 SND 코어가 100%에 가까운지, FW 인스턴스 코어가 포화인지 를 보고, 그 결과에 따라 CoreXL Affinity 튜닝이나 Multi-Queue를 적용할지 판단합니다(상세는 sk101878).

CPU Spike Detective

CPU Spike Detective 는 CPU 사용률이 갑자기 치솟는(spike) 현상을 탐지 합니다. 평상시 모니터링으로는 놓치기 쉬운 순간적 CPU 급증과 그 원인 을 잡아내, 어떤 프로세스·작업이 spike를 일으켰는지 파악하게 돕습니다.

정리하면, CPView로 전반적 자원·코어 상태를 실시간 관찰하고, CPU Spike Detective로 순간 급증을 잡아 , 어디를 튜닝할지 근거를 마련하는 것이 모니터링의 역할입니다. 여기서 본 병목에 따라 SecureXL·CoreXL·Multi-Queue·HyperFlow를 조정합니다.

11 HyperFlow

HyperFlow

HyperFlow(R81.20 이상)는 대용량 연속 연결(elephant flow)을 여러 CPU 코어로 병렬 검사 하는 기술입니다. CoreXL의 한계를 보완합니다.

무엇을 푸는가

Elephant flow 는 바이트 수가 큰 연속 연결 입니다 — 예를 들어 HTTP·HTTPS·FTP·NFS로 큰 파일(리눅스 ISO 등)을 내려받는 연결이죠. 이런 연결은 다른 데이터 세션보다 네트워크 용량을 크게 잡아먹습니다.

문제는 HyperFlow가 없으면 한 elephant 연결을 CoreXL Firewall 인스턴스 하나(코어 하나)로만 검사 한다는 점입니다. 그래서 그 코어의 CPU 사용률이 오르면 처리량이 점점 떨어 집니다. **HyperFlow** 는 하나의 큰 검사 작업을 작은 작업으로 쪼개 여러 코어에 분산 해 이를 해결합니다.

동작과 한계

HyperFlow는 검사 작업(패킷 수신·스트리밍·프로토콜 파싱·패턴 매칭·블레이드 로직 등)을 잘게 나눠 여러 코어로 흘려 , elephant 연결의 처리량을 높이고 응답 시간을 개선합니다. 자동으로 elephant 연결을 감지해 동적으로 코어를 배분 하므로 수동 할당은 필요 없습니다.

켜기 전에 Requirements(요구사항) 를 확인해야 하고, Syntax 로 임계값 등을 설정하며, CPView에서 HyperFlow 동작을 모니터링 합니다. Limitations(한계) 와 Troubleshooting 도 원문에 정리되어 있습니다. Security Gateway 가이드에서 봤듯, HyperFlow는 User Space Firewall(USFW)에서만, 필요할 때만(CPU 여유가 있을 때만) 작동 하며 전체 처리량이 elephant 연결보다 우선합니다.

정리하면, HyperFlow는 한 코어에 묶여 병목이던 대용량 연결을 여러 코어로 쪼개 병렬 검사해, Threat Prevention 같은 무거운 검사가 켜진 상태에서도 대용량 흐름의 성능을 끌어올립니다. 세부 구문·요구사항·한계는 원문 해당 절(sk178070)을 참고하세요.

12 명령줄·커널 참조

명령줄·커널 참조

성능을 깊이 튜닝하다 보면 명령줄과 커널 수준 으로 내려가야 합니다. 이 장은 명령줄 참조, 커널 파라미터, 커널 디버그를 개념 중심으로 정리합니다(방대한 사전은 전용 문서로 넘김).

명령줄 참조

성능 관련 명령 전체는 **R82 CLI Reference Guide** 와 원문 Command Line Reference 절에 정리되어 있습니다. 앞 장들에서 본 핵심 명령을 다시 모으면 — SecureXL의 `fwaccel` / `fwaccel6` , CoreXL의 `fw ctl multik` · `fw ctl affinity` , Multi-Queue의 `mq_mng` / `cpmq` , 모니터링의 `cpview` 입니다.

커널 파라미터

커널 파라미터 는 게이트웨이의 고급 동작을 바꾸는 설정값 으로, 정수(Integer) 또는 문자열(String) 타입입니다(Security Gateway 가이드의 커널 참조와 같은 개념). 성능 튜닝에서는 두 묶음이 중요합니다 — **Firewall Kernel Parameters**(방화벽 동작)와 **SecureXL Kernel Parameters**(가속 동작) 입니다.

적용 방법은 `fw ctl set` 으로 즉석 변경(재부팅 시 사라짐), `$FWDIR/boot/modules/fwkern.conf` 에 적어 영구 적용(재부팅 후 효력) 입니다. 클러스터는 모든 멤버를 같은 값으로, VSX는 모든 Virtual System에 적용, Scalable Platforms는 해당 Security Group 에서 설정합니다. 구체적 파라미터 이름·값은 Check Point Support의 SK 문서나 지원팀 안내 에 따르는 것이 안전합니다.

커널 디버그

커널 디버그 는 게이트웨이가 특정 연결을 어떻게 처리하는지 보여 주는 디버그 메시지를 수집하는 작업으로, 주로 지원팀·R&D가 분석에 씁니다. 흐름은 ① 기본 설정 초기화·버퍼 할당 → ② 디버그 모듈·플래그 설정 → ③ 출력 파일로 수집 시작 → ④ 중지 → ⑤ 복원 입니다. 디버그 필터(Kernel Debug Filters) 로 원하는 트래픽만 좁혀 보고, 연결 수명주기(Connection Life Cycle) 까지 따라가는 절차, 모듈·플래그 목록 도 원문에 정리되어 있습니다.

정리하면, 일상 튜닝은 CPView와 `fwaccel` · `fw ctl` 명령으로 하되, 고급 동작은 커널 파라미터(`fw ctl set` · `fwkern.conf`)로, 깊은 진단은 커널 디버그(`fw ctl debug`)로 내려갑니다. 방대한 명령·파라미터·플래그 사전은 R82 CLI Reference Guide와 원문 해당 절이 담당합니다.