Building Security Into Your DevOps Pipeline

How Next-Generation Firewalls Protect AI Workloads Without Slowing You Down

Gurdeep Kaur Gill — Cisco Systems

CHAPTER 1

The Challenge: Velocity vs Security

Modern DevOps Reality

Release cycles are shrinking dramatically as organizations embrace continuous delivery. Teams now deploy code multiple times per day, fundamentally changing how software reaches production.

Traditional security reviews positioned as "gates" simply don't scale with this deployment frequency. Every manual checkpoint becomes a bottleneck that slows innovation and frustrates engineering teams.

The AI Complexity Factor

AI workloads exponentially increase the risk surface through three vectors: models themselves, training data pipelines, and user prompts.

Unlike traditional applications, threats now target runtime behavior and emergent properties rather than static code vulnerabilities.

> **The imperative:** continuous validation and protection without creating deployment bottlenecks that undermine business agility.

SECURITY LANDSCAPE

Why AI Systems Change Everything

Unstructured Input Ambiguity

AI systems accept unstructured inputs—text, images, audio—that introduce inherent ambiguity. Traditional validation rules designed for structured data fail.

Non-Deterministic Outputs

Unlike conventional software with predictable outputs, AI models generate probabilistic responses. This complicates testing and establishing normal behavior patterns.

Data Provenance Criticality

Training data poisoning attacks introduce subtle biases or backdoors that may not surface until months after deployment.

High-Value Intellectual Property

Models represent months of computational investment and competitive advantage. Model theft through extraction attacks poses unique risks.

Novel Attack Classes

Adversarial inputs exploit model blind spots, prompt injection overrides instructions, membership inference leaks training data.

TECHNOLOGY DEEP DIVE

What NGFWs Add Beyond Legacy Firewalls

Layer-7 Application Awareness

Understands application protocols and context, not just IP addresses and ports. Enables granular control over API endpoints, database queries, and microservice communication patterns.

Payload & Content Inspection

Examines packet contents to detect threats within application data streams. Identifies malicious payloads, data exfiltration attempts, and policy violations in real-time traffic flows.

TLS Inspection

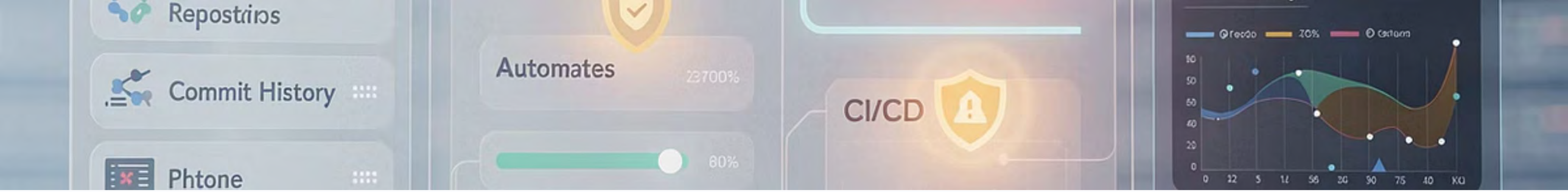
Decrypts, inspects, and re-encrypts traffic to maintain visibility into encrypted communications. Critical for identifying threats hiding in HTTPS and other encrypted protocols.

Behavioral Anomaly Detection

Establishes baselines and identifies deviations from normal traffic patterns. Detects zero-day threats and insider activity through statistical analysis and machine learning models

Identity Context Integration

Enforces policies based on service identity, workload roles, and user context rather than just network location. Essential for ephemeral container and serverless environments



⚠ CRITICAL GAP

What "Shift Left" Misses

The "shift left" movement successfully embedded security into development phases. CI/CD scanners catch vulnerabilities in code, dependencies, and infrastructure configurations before deployment. This is necessary but **insufficient**.

Lateral Movement

Between microservices

Data Exfiltration

Through legitimate channels

Privilege Escalation

In orchestration layers

Time-Delayed Attacks

Triggered by specific conditions

The solution: "shift-left + shield-right" where NGFWs become the continuous control plane enforcing policy at runtime.

FRAMEWORK

Comprehensive NGFW Integration

A holistic approach to embedding next-generation firewall controls throughout the development lifecycle and production runtime environment. This framework transforms security from an external gate into an integrated capability.

Identity-Based Access Control

Workload and service identity integration replacing brittle IP-based rules in dynamic environments.

Policy-as-Code with CI Validation

Version-controlled security policies validated during continuous integration, deployed atomically with application code.

AI-Specific Threat Detection

Custom rules and behavioral baselines tailored to model serving, training pipelines, and inference APIs.

Encrypted Traffic Inspection Strategy

Selective TLS decryption balanced with performance requirements and compliance constraints.

Alert Optimization & Aggregation

Context-enriched incident narratives that reduce false positives and enable rapid response.

Deployment Models: Inline vs Out-of-Band

Choosing the right deployment model for NGFWs requires balancing enforcement capability, operational risk, and performance requirements. Most successful implementations follow a phased approach.

Inline Enforcement

NGFWs sit directly in the traffic path with full enforcement authority. Can block, allow, or modify traffic in real-time based on policy decisions.

Advantages: Complete protection, immediate threat response, policy enforcement guarantee

Requirements: High-availability design, latency optimization, capacity planning for peak loads, failover strategies

Risk: Becomes critical path component; misconfigurations can cause outages

Out-of-Band Monitoring

Traffic is mirrored to NGFWs for inspection and analysis without enforcement capability. Detects threats and establishes behavioral baselines without production impact.

Advantages: Zero service impact, safe learning environment, perfect for tuning and validation

Limitations: Cannot block threats in real-time, relies on reactive response processes

Use cases: Initial deployment, high-risk changes, baseline establishment, new threat detection rules

Recommended approach: Hybrid rollout strategy starting out-of-band to learn traffic patterns and tune policies, then gradually transitioning to inline enforcement for critical paths as confidence builds.

MODERN ACCESS CONTROL

Identity-Based Access for Dynamic Environments

Traditional IP-based firewall rules fundamentally fail in modern cloud-native environments where workloads are ephemeral, containers are transient, and IP addresses change continuously. **Identity becomes the new perimeter** in Kubernetes clusters, serverless functions, and auto-scaling groups where network location is meaningless.

Service Mesh Identity

Mutual TLS (mTLS) certificates issued by mesh control planes like Istio or Linkerd. Provides cryptographic proof of service identity with automatic rotation.

API Gateway Identity

OAuth tokens, mutual TLS certificates, or API keys validated at the gateway layer. Enables centralized authentication and authorization decisions.

Cloud Workload Identity

IAM roles (AWS), service accounts (GCP), or managed identities (Azure) that bind directly to compute resources without credential management.

Policy-as-Code in CI/CD Pipelines

Security policies should follow the same software engineering practices as application code: version control, automated testing, peer review, and continuous deployment. This approach eliminates configuration drift and ensures security keeps pace with application changes.

Version Control

Store firewall and security policies alongside application code in Git repositories. Track changes, maintain history, enable rollback to known-good states.

CI Validation

Automated policy testing during continuous integration: syntax validation, conflict detection, forbidden pattern checks, compliance rule verification.

Synchronized Deployment

Deploy security policies with infrastructure-as-code tools (Terraform, Helm, CloudFormation) atomically with application deployments.

Coordinated Rollback

Application version rollback automatically triggers corresponding policy rollback, maintaining consistency between code and security controls.

This approach enforces **consistency** across environments, provides complete **auditability** of security changes, and ensures **repeatability** for disaster recovery and multi-region deployments.

AI THREAT DEFENSE

Protecting AI APIs: Model Extraction Attacks

Model extraction attacks represent a unique threat to AI systems where adversaries systematically query models to reconstruct their functionality without access to training data or architecture. These attacks can replicate months of computational investment through clever API interactions.

1

Rate Limiting

Per-identity throttling prevents automated extraction through volume restrictions

2

Pattern Detection

Identify abnormal query sequences and suspicious exploration patterns

3

Automation Blocking

Detect and block bot signatures and scripted client behaviors

4

Output Sanitization

Strip or limit probability scores to reduce information leakage

Common Extraction Patterns

- High-volume probing: Thousands of queries mapping input-output relationships
- Boundary exploration: Systematic testing of decision boundaries and edge cases
- Transfer learning abuse: Using stolen model knowledge to train derivative models
- Confidence score exploitation: Leveraging probability outputs to accelerate extraction

Protecting LLMs: Prompt Injection Defense

Large Language Models face a unique attack class where malicious users craft prompts designed to override system instructions, exfiltrate confidential information, or manipulate model behavior. These attacks exploit the models' instruction-following capabilities against themselves.

Prompt Sanitization

Deploy heuristics and pattern-matching rules to identify and neutralize injection attempts. Detect instruction override patterns, delimiter confusion tactics, and role-switching commands before they reach the model.

Request Baselining

Establish expected request structures per endpoint and model type. Flag deviations in prompt length, format, character distributions, and linguistic patterns that suggest adversarial crafting.

Output Monitoring

Inspect model responses for policy violations where technically and legally feasible. Detect credential leakage, system prompt disclosure, and jailbreak success indicators in generated content.

Behavioral Rate Limiting

Track rejection rates and jailbreak-like traffic patterns per client identity. Throttle or block users exhibiting persistent adversarial behavior through graduated enforcement.

Defense-in-depth approach: NGFW controls provide the network enforcement layer, but must combine with application-side guardrails, input validation libraries, and model-level safety training for comprehensive protection.

PERFORMANCE ENGINEERING

Encrypted Traffic Inspection at Scale

TLS encryption protects data in transit but creates a visibility gap for security tools. Without a decryption strategy, threats hide in encrypted traffic flows that constitute over 90% of modern application communications.

Hardware Acceleration

Leverage ASIC and FPGA-based crypto offload to handle decryption at line rate without becoming a bottleneck. Essential for multi-gigabit throughput requirements.

Selective Decryption

Implement risk-based scoping to decrypt only traffic that warrants inspection. Bypass known-safe applications, internal mesh traffic, and compliance-restricted flows.

Session Optimization

Support TLS session resumption to reduce handshake overhead and improve performance. Cache session tickets and enable connection reuse where possible.

Certificate Pinning

Applications using certificate pinning will reject NGFW-issued certificates during inspection. Implement exemption policies for pinned applications or use out-of-band monitoring.

> The firewall cannot become the bottleneck. Size NGFW capacity to handle peak encrypted traffic loads without latency degradation.

FUTURE READINESS

PQC Readiness

Post-quantum crypto: prep for NIST migration + “harvest now, decrypt later”.

Harvest-now risk

RSA and ECC vulnerable to Shor's algorithm on quantum computers. "Harvest now, decrypt later" attacks capture encrypted data for future breaking.

NIST PQC

Post-quantum algorithms published: CRYSTALS-Kyber (encryption), CRYSTALS-Dilithium and others (signatures). Industry implementation beginning.
.

Hybrid TLS

Transition period using both classical and post-quantum algorithms simultaneously. Maintains backward compatibility while building quantum resistance.

Keys + perf

Performance planning for larger key sizes (3-10x current sizes). Increased computational overhead for key operations

> Start with long-lived sensitive data flows; expand as performance is validated.

OPERATIONS

Reducing False Positives & Fatigue

Make alerts actionable: baseline, score, aggregate, add context.

Baseline First

Deploy out-of-band initially to learn normal traffic patterns without enforcement pressure. Establish statistical models of expected behavior across different times, days, and release cycles.
.

Anomaly Scoring

Replace binary alerts with anomaly scoring systems. Generate alerts only when deviations exceed statistically significant thresholds. Reduces noise from minor fluctuations.

Alert Aggregate

Group related events into incidents rather than flooding analysts with individual alerts. Correlate repeated attempts from the same source into coherent attack narratives.
.

Context Enrichment

Enhance alerts with actionable intelligence: service identity, asset criticality scores, destination reputation, recent change history, related incidents. Enable rapid triage decisions.

> Goal: fewer, high-confidence alerts people act on quickly.

IMPLEMENTATION GUIDE

Implementation Roadmap & Key Takeaways

Successful NGFW integration requires a phased deployment approach to balance risk and operational learning.

1

Phase 1: Out-of-Band Monitoring

Deploy NGFWs in monitoring mode to establish baselines and validate capacity without enforcement risk.

2

Phase 2: Policy-as-Code & Selective Enforcement

Migrate security policies to version control and integrate with CI/CD. Begin inline enforcement for non-critical services.

3

Phase 3: AI-Specific Detections

Implement AI-specific defenses like model extraction prevention and prompt injection defenses. Sandbox unknown AI traffic patterns.

4

Phase 4: Performance Optimization & PQC

Optimize TLS inspection throughput with hardware acceleration and pilot post-quantum cryptography (PQC) for production performance.

Key Takeaways

- **Security as Automation:** Implement continuous, automated security with policy-as-code and identity-based controls.
- **Runtime Protection:** NGFWs provide essential "shield-right" capabilities against runtime behaviors exploited in production.
- **AI-Specific Controls:** AI workloads introduce new attack surfaces requiring specialized security measures.

Thank You!

Questions?

Gurdeep Kaur Gill | Cisco Systems | Conf42 DevOps 2026
https://www.linkedin.com/in/gurdeep-gill-20156321/