FROM CODE DEPENDENCIES TO MODEL DEPENDENCIES — A NEW ATTACK SURFACE

Securing the AI Supply Chain: Model and Data Provenance

Executive briefing — Security and ML platform governance leaders | ~20 min

The AI Supply Chain Is a New Dependency Graph

- Pretrained foundation models, third-party fine-tunes, adapters, and training datasets now sit alongside traditional code libraries in the dependency graph
- Each artifact is pulled from external hubs, registries, or vendor APIs with varying levels of trust and traceability
- Unlike source code, model weights are opaque binaries — inspection and diffing tools are far less mature
- A compromise anywhere upstream (base model, dataset, or fine-tuning pipeline) can propagate silently into production systems
- Governance frameworks built for software supply chains do not automatically cover this graph and need deliberate extension

Risk: Downloading Models Without Provenance Verification

- Public model hubs allow open uploads; naming similarity to reputable models does not imply equivalent vetting
- Malicious or tampered weights can embed backdoors that activate only on specific triggers, evading standard testing
- Serialization formats used by some frameworks have historically permitted arbitrary code execution on load
- Without a verified chain of custody, teams cannot confirm a downloaded model matches what its card or documentation describes
- Industry-reported range: a meaningful share of organizations using open-source models report no formal verification step before deployment

Training Data Provenance and Poisoning Risk

- Third-party and web-scraped datasets may contain mislabeled, biased, or deliberately poisoned samples designed to induce specific model behaviors
- Poisoning can be sparse — a small fraction of manipulated samples can measurably shift model behavior on targeted inputs
- Data lineage is often undocumented past the first hop, making it hard to know what a dataset itself was built from
- Fine-tuning and RAG pipelines introduce fresh data-ingestion points that bypass original model vetting
- Provenance tracking must extend to any data used for continued training, alignment, or retrieval augmentation, not just the base corpus

Model Cards and Dataset Documentation as a Governance Control

- Model cards and dataset datasheets are the closest analogue to a software changelog — intended use, training data sources, known limitations, evaluation results
- Treat incomplete or missing documentation as a risk signal, not an administrative gap
- Require documentation as a gating condition before a model can move from evaluation to approved-for-use status
- Standardize an internal template so cards are comparable across vendors and open-source sources
- Documentation should be versioned alongside the artifact it describes, not maintained separately

Extending SBOM Concepts to Models: The Model BOM

- A Model Bill of Materials (Model BOM) inventories base model, fine-tuning data, adapters, libraries, and runtime dependencies for a deployed system
- Captures version, source, license, and cryptographic hash for each component in the model's build chain
- Enables rapid impact assessment when a vulnerability or compromise is disclosed upstream
- Emerging standards (e.g., extensions to SPDX and CycloneDX for ML artifacts) are still maturing — expect format churn
- Model BOM generation should be automated as part of the ML pipeline, not produced manually after the fact

Dependency and Library Vulnerability Scanning for ML Pipelines

- ML pipelines depend on a large surface of Python packages, CUDA libraries, and serving frameworks, each with its own vulnerability history
- Standard software composition analysis (SCA) tools should be extended to cover ML-specific libraries and container images
- Pin and hash-verify dependencies in training and inference environments rather than resolving to latest at build time
- Scan third-party inference servers and model-serving frameworks with the same rigor applied to web application dependencies
- Establish a recurring cadence for rescanning deployed model environments as new CVEs are disclosed

Vetting Third-Party Fine-Tunes and Adapters

- Fine-tunes and lightweight adapters (e.g., LoRA-style) inherit all risks of the base model plus risks introduced by the fine-tuning data and process
- A fine-tune from an unverified source can alter model behavior in narrow, hard-to-detect ways while passing general benchmarks
- Require disclosure of base model identity, fine-tuning dataset sources, and training methodology before approval
- Run targeted behavioral testing on fine-tunes, not just aggregate accuracy metrics, to catch narrow behavioral shifts
- Maintain an approved registry of vetted fine-tunes rather than allowing ad hoc adoption by individual teams

Signing and Integrity Verification for Model Artifacts

- Cryptographic signing of model weights and configuration files allows downstream consumers to verify the artifact has not been altered since publication
- Hash-verify every model file against a trusted manifest at download time and again before each deployment
- Integrate signature verification into CI/CD gates for model deployment, mirroring code-signing practices for software releases
- Where publisher signing is unavailable, establish an internal re-signing step after your own security review
- Store trusted hashes and signing keys separately from the model registry to prevent single-point tampering

Vendor Risk Assessment for AI Platform Providers

- Extend existing third-party risk assessment questionnaires to cover model training practices, data sourcing, and update/rollback procedures
- Ask vendors directly about their own supply chain: do they verify upstream models and datasets they build on
- Evaluate incident notification commitments — will the vendor tell you promptly if an upstream model or dataset is found compromised
- Assess data handling for any customer data sent to hosted inference or fine-tuning services
- Include AI-specific clauses in contracts: audit rights, provenance documentation requirements, and breach notification timelines

Incident Response for a Compromised Upstream Model

- Maintain an inventory (via Model BOM) that lets you identify every system using a given model or dataset version within hours, not weeks
- Define a rollback path to a known-good model version as a standard incident response playbook, tested before it's needed
- Establish criteria for when to disable a model in production versus monitor with heightened logging
- Coordinate disclosure and remediation timelines with the model provider and, where relevant, downstream customers of your own products
- Run a tabletop exercise simulating a compromised base model to validate the playbook before a real incident

Illustrative Scenario: A Compromised Fine-Tune in Production

- Illustrative scenario, not a verified case study — presented to ground the discussion in a plausible operational pattern
- A team adopts a community fine-tune for a customer support assistant without a documented data lineage review
- Weeks later, targeted inputs are found to trigger inconsistent outputs traceable to unvetted training data in the fine-tune
- Root cause analysis is slowed because no Model BOM exists to identify which production systems used the affected fine-tune
- Recovery requires manual audit of every deployment — the scenario a Model BOM and signing process is designed to prevent

A Practical Governance Roadmap

- Phase 1 (0–3 months): inventory current models and datasets in use; stand up Model BOM generation for new deployments
- Phase 2 (3–6 months): require model cards and signed artifacts as a gate for new model approvals
- Phase 3 (6–12 months): extend SCA tooling to ML pipelines; formalize vendor risk questionnaires for AI providers
- Phase 4 (ongoing): run incident response tabletop exercises and refresh the approved fine-tune registry on a fixed cadence
- Sequence by risk exposure — prioritize customer-facing and data-sensitive systems before internal tooling

Organizational Ownership and Skills

- AI supply chain risk spans security, ML platform engineering, legal, and procurement — no single function owns it end to end today in most organizations
- Security teams generally need training in ML artifact formats and model-specific risk; ML teams generally need training in supply chain security practices already familiar from software
- Procurement processes should be updated to route AI model and dataset purchases through the same vendor risk workflow as any other software acquisition
- A named accountable owner for AI supply chain governance prevents the responsibility from being assumed by everyone and owned by no one
- Cross-functional review at model approval time is the practical mechanism that makes this ownership real, not just a policy statement

Next Steps and the Ask

- Approve a cross-functional working group (security, ML platform, legal) to own AI supply chain governance
- Fund a pilot Model BOM implementation on two to three production model deployments within the next quarter
- Mandate model card and signed-artifact requirements for all new model approvals starting next cycle
- Commission an initial vendor risk assessment pass for the top AI platform providers currently in use
- Set a 90-day checkpoint to report inventory coverage, gaps identified, and adjusted timeline to full rollout