Edge AI Software Supply Chain Visibility

By Alexandra Tzoumas

Edge AI Software Supply Chain Visibility

By Alexandra Tzoumas

White Paper | Edge AI for National Security | July 2026

Abstract

Organizations are deploying artificial intelligence (AI) far faster than they can document its underlying components. Frameworks for recording a model's provenance exist, covering training data, licensing, and modification history, but there is no unified standard required for edge and defense procurement. The result is a set of conditions that accumulate during normal operations: documentation practices remain optional and uneven, disclosure across the vendor-to-government boundary is not mandatory, and the resulting gaps in visibility widen when no immediate crisis forces them into view. This paper describes the components of the AI software supply chain, the points at which visibility is lost, and the national security exposure that accumulates when those components are deployed without a documented provenance record. It surveys the standards and tooling that address parts of the problem and describes where their coverage stops short, and it outlines the elements a standardized AI Bill of Materials (AIBOM) would need to capture to close the gap.

Executive Summary

For conventional software, knowing what is inside a system before you field it is routine, supported by mature standards and tooling. AI does not yet clear the same bar, and the gap is widest precisely where the stakes are highest: at the edge, on the devices carried into contested environments for military and government use. The shortfall has two layers. Part of it is structural and permanent: a fielded AI system is rarely built from scratch, and once a foundation model is trained, its history is sealed inside it, because the training data cannot be read back out and the weights are not human-readable. The rest is a matter of practice and disclosure: the schemas that could document a model's trackable history, its training data sources, licensing, and modification history, exist but are neither uniformly followed nor required, and where a developer has done the work, there is no standing expectation that it be shared with the government that fields the system. The visibility that supply chain security takes for granted, the ability to trace where a component came from and what it carries, is therefore partly unavailable by construction and partly just undocumented. When the National Security Agency (NSA), Cybersecurity and Infrastructure Security Agency (CISA), and the Federal Bureau of Investigation (FBI) issued joint guidance in May 2025, they went for a single word to describe modern training data: opacity.

That word is the problem this report takes up, and the findings below are why it is worth reading in full.

The opacity reaches the biggest names in AI, not just the obscure ones. Two computer vision models well suited to edge deployment, Florence-2-large from Microsoft and OWLv2 from Google, are downloaded at scale and already appear in defense-adjacent use. Both arrive with training histories their own publishers never fully disclose. An evaluator reviewing one for a defense application is being asked to trust it on the strength of a dataset's name and a partial description of what went into it.

The same opacity that makes these models hard to audit makes them cheap to corrupt. Researchers have shown that poisoning a web-scale dataset can cost as little as tens to a few thousand dollars, with a manipulated fraction well below one percent enough to take effect. The property that hides the problem is the property that attracts it.

For government use, the worst case is not the model that fails the test. It is the one that passes yet that is compromised in ways that were not discovered prior to fielding. A compromised foundation model has no patch. On the account of one interviewee, its discovery inside a government system does not trigger a fix; it triggers a national security investigation. The failure mode that should worry an evaluator is not the poisoned model caught on the bench and discarded, but the one whose compromise surfaces only after it has been operating inside a classified environment.

The edge turns every one of these from a hard problem into a compounded one. Post-inspection, optimization, and quantization transform the model, producing a runtime artifact that diverges from the one originally reviewed. The fielded device is hard to reach for re-verification.

Updates become an operational problem rather than a routine one, and a model can degrade silently while the path to correct it is cut off.

Underneath these findings is a timing argument that shapes this entire report. The same competitive pressure that pushes organizations to adopt AI quickly is what pushes the question of provenance out of view, and with no standard requiring that history be documented, the risk accumulates quietly during peacetime, when nothing in the moment forces it into the open, the same way a company will keep shipping software built on unvetted open-source dependencies for years, until a breach or an audit forces a reckoning with what's actually inside the product. Drawing on interviews conducted, this report contrasts an entity operating under existential urgency, which accepts elevated supply chain risk because it has no choice, with one that still holds the option to set a high bar before that pressure arrives. Peacetime is not presented as a warning that time is short. It is presented as the moment that offers the clearest view.

The report is scoped to edge AI software, meaning inference and deployment rather than training infrastructure, and traces the software side of the supply chain rather than the hardware beneath it. It takes an edge AI system apart component by component to show where visibility is held and where it is lost, then sets out seven core risk areas: Data Provenance, Foundation Model Cascading Risk, Reproducibility and Verification, Stack and Pipeline Fragility, Permissive Licensing and Deployment Gap, Geopolitical Risk, and the Policy Vacuum. The first four concern what cannot be seen about a model in motion: what data it learned from, what it inherited from the models beneath it, whether it can be independently rebuilt and verified, and what happens to it as it is transformed for the edge and updated in the field. The remaining three are different in kind. They turn on intention, leverage, and accountability: questions no technical assessment can uncover.

The report does not claim that full visibility is achievable. The data cannot be fully reconstructed, the weights cannot be fully audited, and the dependency chain cannot be fully traced, because that is how these systems are built. What it offers instead is a way to narrow that uncertainty rather than remove it: the structural opacity is permanent, but the surrounding margin, what can be documented about a model's sources, licensing, and modification history, can be brought within manageable bounds. The point is to reduce what is unknown to a level a decision-maker can weigh, so that the risk is carried knowingly rather than inherited blind. That difference is the one part of the problem available to act on now, without waiting for a standard that does not yet exist. The report describes the dynamic and leaves the conclusion to the reader.

1. Introduction

As AI is increasingly deployed on edge devices for military and government use, how should developers approach visibility across the AI supply chain to ensure resilience before crisis forces the conversation?

That is the question this paper takes up. It is not a technical manual, and for most of its length it describes conditions rather than prescribing fixes. Where it does arrive at recommendations, in the closing section, they follow from what the description makes visible. It is a story built on facts, meant to help the people who build, acquire, and field these systems think clearly about a risk that is easy to miss precisely because nothing in the moment forces it into view.

What the paper does is narrow and specific. It documents the visibility gap in the edge AI software supply chain, explains where that gap comes from, and uses the contrast between peacetime and wartime as a lens for why the timing matters. While it does not tell the reader what to conclude, it is written for three audiences at once: the government program and acquisition staff who evaluate these systems, the business and operations leaders who set the pace of adoption, and the developers who build them. It is intended to help each of them recognize key visibility gaps, assess where their own practices may be exposed, and consider where deliberate attention or policy may be required. Rather than prescribing a solution, the paper aims to provide readers with a sharper lens to identify, weigh, and act on supply chain visibility challenges appropriate to their own roles and contexts. It describes a dynamic and leaves the conclusion to them.

Two ideas run underneath the whole account, and it is worth naming them at the outset. The first is that visibility is the precondition for everything else. You cannot secure, verify, or make a sound risk decision about what you cannot see. Evaluation and governance both depend on it, which means supply chain visibility is not a separate lane from those efforts but the ground they stand on. The second is that the gap in visibility is structural, not a failure of effort. A model's training data cannot be read back out of the finished model, and its weights are not human-readable. When a government security body such as the National Security Agency (NSA) or Cybersecurity and Infrastructure Security Agency (CISA) describes modern training data with the word opacity, it is confirming that the gap is real and built in, not a matter of someone not trying hard enough.

Those two ideas frame why peacetime is the moment this examination is easiest. The same conditions that make the supply chain hard to see are also the conditions under which it can be examined without a crisis dictating the terms. A dependency that accumulates one component at a time is easy to trace after the fact and nearly impossible to see while it is forming. The purpose of what follows is to make it visible now, while there is still slack in the system to act on what the map shows. The purpose of what follows is to make it legible now, while there is still slack in the system to act on what the map shows.

Intended Audience

The paper is written for three audiences at once, each of which encounters the visibility gap from a different angle.

For government program offices and acquisition staff, the paper describes a gap that becomes a procurement concern the moment a model is evaluated for a defense application. It is meant to help acquisition professionals recognize where a model under review may carry undocumented history, and to ask sharper questions before a system is fielded.

For business and operations leaders, it traces where that gap comes from and why the pressure to move quickly tends to push it out of sight. It is intended to make visible the organizational dynamics of competitive pace, cost pressure, and adoption fever that let supply chain risk accumulate quietly during normal operations.

For developers, it walks through the assembly of an edge AI system component by component and shows where visibility is lost along the way. It is meant to be concrete about the specific points in the build and deployment pipeline where provenance breaks down.

The work is intended as groundwork for the Edge AI Foundation Defense Working Group's governance recommendations.

2. Why Now: The Urgency Argument

By the autumn of 2023, Ukraine's drone war had run into a supply problem. The People's Republic of China (PRC) companies had begun cutting back sales of drones and components to Ukrainian buyers, and the firms still willing to sell often required buyers to route purchases through complicated networks of intermediaries. New Chinese export rules on drone components came into force on September 1, 2023, and many feared they would worsen Ukraine's supply difficulties heading into the winter. At the time, Ukraine was losing an estimated 10,000 drones a month.[1] The dependency underneath this was not incidental: by later estimates, up to 97 percent of the components used in Ukrainian-made drones still came from China.[2]

The reliance had not been chosen so much as inherited. It accumulated one component at a time: flight controllers, motors, sensors, and the rest, each sourced on the merits of cost and availability, none of them flagged at the moment of purchase as a strategic vulnerability. The dependency became visible only when it became a lever, when the supplier began deciding who could buy and who could not. Ukraine had built its most vital battlefield capability on a supply chain it did not fully see until an adversary chose to constrain it.

The pattern is worth naming precisely, because it recurs. No single decision created the exposure. Nobody mapped the chain while there was still slack in the system to act on what the map showed. The vulnerability was legible in hindsight and invisible in the moment, not because anyone was careless, but because the conditions that would have made it visible never arrived until the pressure did.

A comparable dynamic is taking shape in AI software supply chains, and the mechanism driving it is competitive rather than adversarial. Organizations adopting AI are moving quickly, and the speed is itself a form of risk avoidance: the perceived cost of falling behind a competitor or a peer institution outweighs, in the moment, the diffuse and unquantified cost of not knowing exactly what went into a given model. The same posture that looks like prudence, keeping pace and not ceding ground, is what pushes questions of provenance, licensing, and modification history out of view. What gets pushed out of view is visibility itself, and visibility is the precondition for every judgment that comes after it: you cannot secure, verify, or make a sound risk decision about what you cannot see. This dynamic has a specific current example. Most edge AI systems are not built from scratch; they are adapted from foundation models trained by someone else, and a growing share of the low-cost foundation models drawing attention are released by Chinese developers. When the cheapest capable starting point is a model whose training data and provenance are undisclosed, the speed-versus-visibility tradeoff stops being

[1] Paul Mozur and Valerie Hopkins, "Ukraine's War of Drones Runs Into an Obstacle: China," *New York Times*, September 30, 2023. The 10,000-per-month loss figure is attributed in the article to the Royal United Services Institute (RUSI).
[2] "China's Drone War in Ukraine," *The Diplomat*, January 2026. The article states that up to 97 percent of components used in Ukrainian-made drones are of Chinese origin, presented as the outlet's reported estimate.

abstract. The question of what went into a given model becomes a question of whose model it was, and that is a thread this paper picks up in detail later.

Pace compounds this on its own. When a new AI model or fine-tuned variant appears every few weeks, there is no natural interval in which to audit any single one fully. By the time a provenance review of one version would conclude, later versions have already shipped, and the operational pressure has shifted from examining what was adopted to keeping current with what is available. The absence of a pause is not a lapse in discipline; it is a structural feature of the release cadence.

There is a second, AI-specific reason the question is pressed now, one that runs opposite to the intuition that visibility improves as a technology matures. Project Maven, the Department of Defense AI program established to speed up machine learning and data integration for military intelligence, predated the frontier-model era, and the government held substantial visibility into vendor source code as a condition of the contracting relationship. As Lieutenant General John "Jack" N.T. Shanahan, USAF (Ret.), former Director of the Joint Artificial Intelligence Center (JAIC), put it, that visibility existed "because it was kind of a condition for these companies being able to work with us in the first place."[3] Data provenance was clear by construction, because the government supplied its own data to turn a vendor's algorithm into a working model. In Shanahan's account, the company's algorithm "was really nothing to us until we used our own data from combat operations, from drone video," to build a performing model; whatever the vendor had trained on beforehand mattered comparatively little. With frontier models, that visibility has dropped sharply. Shanahan described a lack of visibility spanning data provenance, how the models are trained, and what the weights and biases are: "the government isn't getting a lot of visibility on that." Visibility, in other words, did not merely fail to keep pace with capability. It regressed as the technology advanced. This is not a matter of insufficient effort but of how the models are built: a model's training data cannot be read back out of the finished model, and its weights are not human-readable, so the opacity is structural.

The distinguishing feature of the present moment is that the reckoning has not been forced. Ukraine had no choice but to accept elevated risk under existential pressure; its standards were set by urgency, and the processes it skipped are ones it now has to reconstruct after the fact. Shanahan drew a similar contrast, noting that entities like Ukraine must accept higher risk under existential urgency. At the same time, the United States retains the option to set a high bar for evaluation before that pressure arrives. The terms are used here in a specific sense. Wartime describes operating under existential or immediate operational pressure, where a system must be fielded whether or not its supply chain can be fully accounted for, as Ukraine has had to do. Peacetime describes the condition in which that pressure has not yet arrived, and an organization still holds the option to examine what it is fielding before a crisis dictates the terms. The peacetime condition is the inverse of Ukraine's: the conditions that make the supply chain hard to see are also the conditions under which it can be examined without a crisis dictating the terms. As Shanahan put it, if the United States does not establish what right looks like during model training, testing, and inference in the absence of adversarial attack, it will pay for that failure once dedicated adversarial attacks corrupt fielded systems.

[3] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.

That observation is the reason this paper exists. Not as a claim that time is short, but as a statement about which moment offers the clearest view. And that view is hardest to hold at the edge, for reasons the sections that follow make concrete.

3. Defining Visibility

Managing and securing a software supply chain relies on being able to trace where each component came from, whether it is intact, and what it depends on. This is normal, mature practice for software, and visibility is the precondition for every security control and mitigation plan built on top of it: you cannot secure, verify, or make a sound risk decision about what you cannot see. In a functioning software supply chain, that means tracing the origin, integrity, and dependencies of each component, and maintaining chain of custody over updates and modifications across the full deployment lifecycle.

Today, AI does not clear that bar the way ordinary software does. Visibility into the model is not merely hard; it is structurally denied in a way it is not for other software, and the two halves of the model resist inspection for different reasons. The weights are the first. They encode everything the model learned, but not in any human-readable form. As Jan Ernst, Director of AI at Latent AI, describes it, a modern foundation model has so many degrees of freedom that it becomes mathematically hard to understand what it is doing; direct interpretability is, for practical purposes, out of reach. The only way to say anything about the weights is to test how the model behaves and infer the rest, which Ernst characterizes as probing the model and hoping the tests are representative rather than reading the answer out directly.

The training data is the second, and it is a different kind of problem. It cannot be probed at all, because the data does not survive as data inside the finished model. It is absorbed into the weights during training and cannot be read back out. So the only way to say anything about what a model was trained on is, again, to test what it does, not to inspect what went in. The consequence is that even a perfectly documented bill of materials (BOM) eventually reaches one component, the model itself, that resists the very inspection visibility required. The other components in the assembly, the ones that surround the model, are more tractable, and the next section takes the assembly apart piece by piece to show where each kind of visibility is held and where it is lost.

This is not only our characterization. The joint guidance issued by the NSA, CISA, and the FBI in May 2025 uses the word opacity to describe web-scale training datasets, noting that their scale and opacity make them especially difficult to audit or govern.[4] Opacity is visibility's direct opposite. When a government security body names opacity as the defining condition of modern AI data, it is confirming that the gap is real and structural, not a matter of insufficient effort.

Why the Edge Is Different

An edge model can look like the simpler case. It is smaller than the frontier systems it often descends from, tuned to one fixed task, and shipped as a single self-contained package rather than a sprawling service running in a data center. A note on scope follows from that descent.

[4] National Security Agency et al., *AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems*, joint cybersecurity information sheet, May 22, 2025.

The systems this paper concerns are the computer-vision and small language models actually fielded at the edge, the object detectors, image-text models, and compact task-specific models that run on a device. Frontier models, the largest general-purpose systems at the capability frontier, enter this account as the upstream ancestors these edge models are distilled, fine-tuned, or adapted from, not as the artifact running in the field. The two are related, because an edge model inherits whatever its frontier ancestor carried, but the visibility problem at the edge is its own problem. Much of the public discussion of AI risk concerns frontier models in data centers; the conditions described here concern what happens to a model after it leaves that setting and is compressed onto hardware carried into the field. The intuition that follows is that a smaller, bounded, single-purpose artifact should be easier to see into. The intuition is worth stating because it is worth correcting. Each of those apparent simplifications is a point where visibility is lost rather than gained. Compression, containerization, and field deployment are not conveniences that make the system easier to inspect. They are the transformations that make it harder. Everything the previous section defined as the precondition- the ability to trace a component's origin, confirm its integrity, account for its dependencies, and hold chain of custody over it through the deployment lifecycle- becomes harder to hold at the edge, not easier. This section sets out where, and why.

None of the differences that follow is unique to the edge in kind. A model in a data center can drift, can be updated without full provenance, can carry undocumented dependencies. The distinction is one of degree and of recoverability: at the edge, each of these conditions is compounded, and the means to detect or correct it is often removed.

The model is transformed after it is inspected. A model does not travel from repository to device unchanged. To run within a device's size, weight, and power (SWaP) limits, it is fine-tuned, modified structurally, then recompiled and quantized for the target hardware. By the time that sequence is complete, the result is, in effect, no longer the same model, even though it still performs its intended task.[5] The weights that come out are not the weights that went in. As Abelardo Lopez noted, the optimization tooling that performs this adaptation, the compilers and linkers, carries dependency chains of its own, so the transformation adds provenance questions rather than resolving them.[6] The consequence is that the model a program office reviews and the model that actually runs on the device are not the same artifact. A cloud-hosted model generally runs in something close to the form it was evaluated in; at the edge, inspection and deployment are separated by a mutation step that is rarely tracked with the rigor applied to the model or its training data.

The fielded device is hard to reach for re-verification. Once a model is deployed to an edge device, the ability to reach back into it, to confirm what is running or verify its state, is often limited by the operating environment. Chris Ritter, Chief Federal Application Architect at Dell Technologies, described the difficulty as no longer one of raw reach. Systems like Starlink can connect nearly anywhere; the problem is electronic warfare. A device that broadcasts for any

[5] The description of Latent AI's optimization process is Shelly Tzoumas's account of the company's workflow. Author interview, July 1, 2026.
[6] Author interview with Abelardo Lopez, Director of Compiler and Embedded Systems, Latent AI, June 17, 2026.

sustained period raises a flag, because an adversary does not need to read the traffic to act on it. In Ritter's framing, "the transmission itself becomes a location."[7] Connectivity at the edge is therefore not merely unreliable; using it can be dangerous. A cloud system is reachable by definition and can be re-inspected continuously. At the edge, the window in which the system can be verified may effectively close at the moment of deployment.

Updates are an operational problem, not a routine one. A fielded model does not stay static; it needs updates to patch vulnerabilities, correct for data drift, and adapt to changing conditions. The May 2025 joint guidance from the NSA, CISA, and FBI treats every point at which new data or feedback enters a model as requiring the same verification and validation as the original training, which makes each update a new supply chain event rather than a maintenance action.[8] In a data center, pushing that update is trivial. At the edge, over jammed or denied connectivity, delivering a verified update is one of the hardest problems in the system, and the delivery mechanism itself becomes a point of exposure. Each update carries its own chain-of-custody questions: who approved it, who built it, who delivered it, whether it was altered in transit, and who was responsible for uploading it and, as necessary, performing field-level final testing; at the edge, those questions may have to be answered without the connectivity to verify the answers fully.

Degradation is silent, and the recovery path may be severed. Even a model left entirely alone degrades. As the operational environment shifts away from the conditions the model was trained on, data drift sets in and accuracy erodes, while the model continues to report confident outputs. In a reachable system, that drift can be detected and corrected. At the edge, in a communications-denied environment, the failure is exposed to a slow decline with no alarm attached: the update chain is severed, and the only symptom is a model quietly becoming wrong. The failure is both invisible and, for the moment, uncorrectable.

The blind spots ship sealed inside one object. For the edge, the model is bundled into a container, a self-contained package holding the model, framework, dependencies, drivers, and base OS image, which unpacks and runs as one system. This is where boundedness is most likely to be mistaken for clarity. The direct contents are mostly visible, but transitive dependencies, the dependencies of those dependencies, remain the primary blind spot. Mark Griffin, Director of Software Architecture at Latent AI, noted that the base OS image, runtime libraries, hardware SDKs, and CI/CD toolchain can each carry vulnerabilities or licensing obligations along with the artifact, and that registries provide versioning without full provenance.[9] Containerization does not resolve the upstream unknowns; it seals them into a single deployable object and ships it to a device that may be hard to reach again. The self-contained package is not a small problem to see into. It is the same set of unresolved histories, gathered into one artifact and closed.

---

[7] Author interview with Chris Ritter, Chief Federal Application Architect, Dell Technologies (Federal), July 14, 2026.
[8] National Security Agency et al., *AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems*, joint cybersecurity information sheet, May 22, 2025.
[9] Author interview with Mark Griffin, Director of Software Architecture, Latent AI, June 18, 2026

Taken together, these are not five separate problems. They are one pattern. The edge lifecycle repeatedly transforms the model and then removes the ability to look back at what it became: optimization changes the artifact before it ships, deployment places it where it cannot easily be reached, and the update path that would allow correction is the first thing a contested environment takes away. The stakes of that pattern are raised further by what edge systems are often for. Autonomy is frequently the point of an edge deployment, which means an undetected transformation or a silent drift can act without a person positioned to catch it. The next section takes the system apart component by component, to show exactly where in that lifecycle each transformation happens and each blind spot forms.

4. Anatomy of the AI Supply Chain

An edge AI model does not run on its own. It sits on top of a stack of software: inference engines, runtimes, drivers, and libraries, each built on the layer below it. Every layer carries its own origin, its own dependencies, its own history. Only when all of them come together can a vision model look at a drone's camera feed and decide what it is seeing, or a small language model read a garbled radio transmission and pull out the call sign that matters. To understand where visibility is lost, it helps to take the assembly apart and inspect each piece.

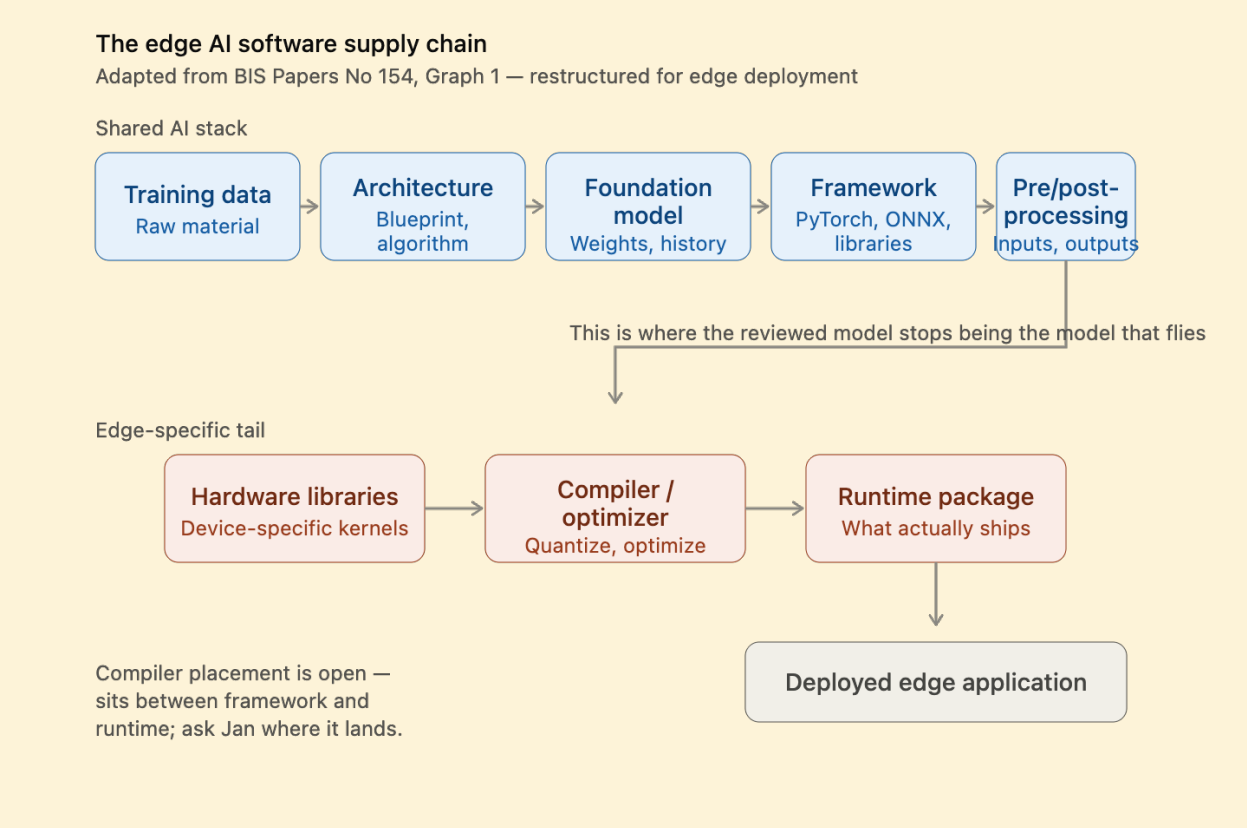

Figure 1. *The edge AI software supply chain, from shared stack through the edge-specific tail. Adapted from Gambacorta and Shreeti, The AI Supply Chain, BIS Papers No. 154, Graph 1; restructured to show edge deployment stages.*

The training data is the material the model learned from, and its origins, along with any ongoing changes, are frequently undocumented even for models from major technology companies. The model architecture is usually inherited rather than designed, and comes with whatever properties it already had; the weights encode everything the model learned and are not human-readable. The foundation model carries the full history of everything that went into it, and everything built on top inherits that history whether the downstream developer knows the dependency exists or not. The algorithm is publicly documented for widely used models. Pre- and post-processing shape what the model sees and what the operator sees. Open-source frameworks, like PyTorch and ONNX, and the libraries beneath them, carry their own dependency tree into deployment.

All of this is true of any AI system, at the edge or in a data center. This paper traces the software side of that assembly. The hardware beneath it, the advanced chips, high-bandwidth memory, and interconnects on which every model ultimately runs, carries its own supply chain, and by most accounts a more geopolitically concentrated one. The manufacture of cutting-edge chips and the processing of the critical minerals they depend on sit in the hands of a small number of firms and countries, and have already become the subject of export controls and national industrial policy. That story is documented in detail elsewhere; the Evolving AI Supply Chain report maps the hardware, compute, and capital layers at length.[10] We note the hardware dependency here to place it, not to trace it, and turn to the parts of the chain that are software.

The Edge-Specific Tail

What makes an edge system different, and harder to see clearly, is the edge-specific tail: the hardware libraries, the compiler and optimizer, and the runtime package. This is the point where the model that was examined diverges from the model that actually flies. Whatever is examined at the point of review is the model before this tail ran; the hardware libraries, the compiler and optimizer, and the runtime package each change the model afterward. Anyone assessing only the model as delivered, without accounting for these three steps, is not looking at what ends up on the device.

Optimization and Compilation

Optimization and compilation take the model as it comes from the ML framework and transform it into an intermediate representation that can be manipulated to produce an artifact with lower latency, a smaller memory footprint (through quantization, for example), or both, usually

[10] Nikhil Mulani, *An Evolving AI Supply Chain* (University of California, Berkeley, October 2025), sections on hardware and computing.

targeting a specific hardware device constrained by size, weight, and power (SWaP) and taking advantage of any capabilities present on it, such as hardware accelerators. Models are often not small enough as-is for edge hardware. At Latent AI, the process was described this way: a foundation model is fine-tuned against a proprietary dataset, sometimes modified structurally, then recompiled and quantized to fit the target device, and by the time all of that is done, it is, in effect, no longer the same model, even though it still performs the intended task.[11] Compilers and quantization tools do this adaptation, and that tooling is itself software with its own origins and dependencies. The process changes the model: the weights that come out are not the weights that went in. The consequence is distinctively an edge concern. Any deployed model can drift from the state in which it was reviewed, through retraining or continued fine-tuning. Still, at the edge the divergence is compounded and harder to recover, because optimization and quantization mutate the model before it ever ships and the fielded device may be difficult to reach for re-verification.

Deployment Container

For the edge, the model is bundled into a container: a self-contained package holding the model, framework, dependencies, drivers, and base OS image, which unpacks and runs as one system. Everything converges here. The direct contents are mostly visible, but transitive dependencies, the dependencies of those dependencies, are the primary blind spot. Griffin noted that the base OS image, runtime libraries, hardware SDKs, and CI/CD toolchain can each carry CVEs or licensing obligations along with the artifact, and that registries such as ECR provide versioning but not full provenance.[12] This is where the upstream blind spots stop being separate problems and become one sealed, deployable object, shipped to a device that may be hard to reach again.

Update and Retraining Pipeline

A fielded model does not stay static. Updates fix vulnerabilities, improve accuracy, and adapt to changing conditions, and the pipeline is how new versions reach a deployed device. Every update is a new supply-chain event; the May 2025 NSA, CISA, and FBI guidance calls for verification and validation each time new data or feedback enters the model.[13] The SWaP-constrained edge makes delivery acutely hard, because pushing a verified update over jammed or denied connectivity is an operational problem rather than a routine one. Each update raises its own chain-of-custody questions: who approved it, who built it, who delivered it, and whether it was altered in transit. Even a model left alone degrades, as data drift erodes accuracy over time. A model clean at deployment can be compromised through a later update, and on a device that is hard to reach, the ability to verify and control that update may be severely limited.

---

[11] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026. and Shelly Tzoumas, July 1, 2026. The characterization of the optimization-and-quantization process is Shelly Tzoumas's description of Latent AI's workflow.

[12] Author interview with Mark Griffin, Director of Software Architecture, Latent AI, June 18, 2026.

[13] National Security Agency et al., *AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems*, joint cybersecurity information sheet, May 22, 2025.

There are real, mature standards for software supply chains, but they inventory the binary and the software around the model, not the model itself. A software bill of materials (SBOM) is an inventory of the software components inside a build; Supply-chain Levels for Software Artifacts (SLSA) is a framework for build integrity and provenance. Chainguard is one of several companies that build tooling and products to help organizations meet SLSA levels. The model's own anatomy can be sorted a different way, into three kinds of SBOMs, and doing so shows exactly where these standards reach and where they stop. As David Aronchick CEO at Embedl put it, existing standards are really a standard for the binary and the pieces that surround the model, not for the model itself.[14]

The Binary BOM covers the software artifact and its surrounding pieces, and is reasonably handled by existing tooling. The Data BOM covers what data went in; Aronchick described data as an "utter black box," not introspectable, where provenance and lineage sit largely untouched by current standards. The Topology BOM covers how the pieces are wired together and deployed, which he described as poorly handled today. Many of the component pieces of an AI system, in his assessment, do not have bills of materials at all right now.

Existing standards inventory the binary well, but coverage stops there. Data and topology, where much of the model's actual behavior and structure is determined, have no equivalent. The gap is not that the tools point at the wrong thing; it is that they do not yet reach far enough. New attempts to close this documentation gap are emerging as the industry evolves, but they are not yet adopted.

[14] Author interview with David Aronchick, CEO Embedl, July 15, 2026.

5. The Core Risk Areas

The previous section took the edge AI system apart to show where visibility is held and where it is lost. This section turns from anatomy to consequence: what those blind spots actually put at risk, and why each one resists the tooling a program office might expect to catch it.

Seven risk areas follow. They are not a ranked list, and they are not independent of one another; several are the same underlying gap seen from a different position in the supply chain. But they sort, roughly, into two kinds of problems. The first four are visibility problems depicting what cannot be seen or proven about a model in motion: what data it learned from, what history it inherited from the models beneath it, whether it can be independently rebuilt and verified, and what happens to it as it is transformed for the edge and updated in the field. The last three are different in kind. They are not gaps that better tooling would close, because what is missing is not information. Permissive licensing, silent foreign dependency, and the policy vacuum are questions of intention, leverage, and accountability, and no instrument reveals them, because they are not technical properties of the artifact.

The order is deliberate. It begins at the model's origin, its training data, and moves outward through the layers built on top of it, then through the pipeline that carries it to the field, and finally out to the market and governance conditions that determine who is accountable for any of it. Each risk is presented with the interview evidence that grounds it and, where one exists, a real-world precedent from software supply chain security that shows the failure mode is not hypothetical. And each closes on the same question the paper keeps returning to: if the visibility to manage this risk is not going to come from the artifact itself, then it has to come from somewhere, and someone has to be responsible for supplying it. For readers seeking actionable insights or decision-making frameworks, key recommendations and approaches for program staff are primarily collected in Sections 8 (Existing Attempts at a Fix) and 9 (Governance & Residual Risk), which offer practical considerations, frameworks, and guiding questions for organizations navigating these risks.

Data Provenance

A model's behavior is shaped entirely by what it was trained on, but that training history is largely unrecoverable once training is complete. Training data origins are frequently unknown or undisclosed, even for models from major technology companies. Because of how AI is built, you cannot read a dataset back out of a finished model: the data is absorbed into the weights during training and does not survive as data. So if the origin of the data was never documented, the visibility gap is permanent. No amount of downstream inspection recovers it, and visibility, the precondition for every judgment an evaluator later has to make, is foreclosed at the source.

This is not a hypothetical concern about obscure models from unknown sources. It is true of computer vision models published by the largest technology companies in the world, downloaded hundreds of thousands of times, and already appearing in defense-adjacent applications.

Consider Florence-2-large, a computer vision model published by Microsoft. It is a capable, mid-sized model at 770 million parameters, small enough to be a candidate for stronger edge hardware, and it performs object detection, dense region captioning, and OCR from simple text prompts. Its model card names the dataset it was trained on: FLD-5B, comprising 5.4 billion annotations across 126 million images. But the name is close to the extent of the disclosure. Where those 126 million images came from, who collected them, who owns them, and what they contain is not meaningfully documented. A developer building on Florence-2 knows the dataset's name and its size, and little else about its origin.

The model is distributed on public repositories such as Hugging Face, with recent monthly downloads reported in the hundreds of thousands. There is no verification of who is downloading it or for what purpose, and no visibility into who has modified it. Its model tree lists dozens of fine-tuned and adapted versions, each a derivative whose own training history is unknown. This raises a question the paper returns to in the cascading-risk section. If someone fine-tuned Florence-2 on adversarial or poisoned data and republished it under a slightly different name, a developer could pull that compromised version into an edge system without knowing it was not the original.

OWLv2, Google's zero-shot object detection model, tells a similar story. At roughly 200 million parameters, it is a stronger candidate for true edge deployment than Florence-2, and its ability to detect objects described in plain text, without prior training on those categories, is exactly the kind of function a fielded military or critical-infrastructure surveillance or targeting system might want. Its model card does name some sources: the CLIP backbone was trained on "publicly available image-caption data" gathered through "a combination of crawling a handful of websites" and pre-existing datasets such as YFCC100M, with the prediction heads fine-tuned on detection datasets such as COCO and OpenImages.[15] The disclosure is not blank, then, but it is partial in the place that matters most: the card acknowledges that a large portion of the training data comes from crawling the internet, without documenting what that portion contains or where it originated.

OWLv2's provenance problem also runs deeper than its own training, because it is built on top of CLIP, OpenAI's foundation model, which is two layers of inherited dependency. CLIP was trained on roughly 400 million internet-scraped image-text pairs, and neither OpenAI nor Google has published a complete accounting of where those images came from, who owns them, or what biases they carry. Whatever provenance gaps exist in CLIP travel silently into OWLv2 and into everything built on OWLv2 in turn.

That same opacity is not only a documentation problem; it is what makes these datasets cheap to attack. Researchers have shown that web-scale datasets can be poisoned for trivial sums: in one study, controlling 0.01 percent of the LAION-400M or COYO-700M dataset would have cost about $60, and poisoning a comparable share across ten popular datasets came to roughly $1,000, with poisoning rates as low as 0.001 percent shown to be effective.[16] The opacity that

[15] OWLv2 model card, Hugging Face (google/owlv2-base-patch16), "Training Data" section.

[16] Nicholas Carlini et al., "Poisoning Web-Scale Training Datasets Is Practical," 2024 IEEE Symposium on Security and Privacy (arXiv:2302.10149).

makes a dataset hard to audit is the same property that makes it inexpensive to corrupt. This is the precise inverse of the visibility this paper argues for, and it is structural rather than a failure of diligence: the joint guidance issued by the NSA, CISA, and the FBI in May 2025 uses the word opacity to describe exactly these web-scale datasets.[17]

The reason this matters beyond documentation hygiene is that undocumented data is not merely unknown; it is manipulable. That same May 2025 joint guidance identifies data supply chain vulnerabilities as a primary risk, warning that organizations relying on third-party data may unknowingly ingest material from untrustworthy sources, including data brokers, open-source datasets, and unvetted vendors, in ways that can degrade model accuracy or introduce legal exposure.[18]

According to Steve Kenyon, Lead AI Research Engineer at Latent AI, the underlying obstacle is reproducibility. Because training data is almost always proprietary, a downstream user has no way to verify what actually went in, and therefore no way to independently confirm the data was clean.[19] Verification depends on access the downstream user does not have. Kenyon was also clear about the limits of the standard tooling here: generating a software bill of materials does not detect poisoned data in a foundation model, because an SBOM inventories the surrounding software, not the model's learned behavior. The only reliable way to surface a problem is evaluation, testing what the model does.

That sequence carries a consequence worth stating plainly, because it is where the risk concentrates for government use. Discovering a poisoned model and discarding it is the good outcome. In Kenyon's account, there is no "fix" for a compromised foundation model; it is removed from use, and in a government context its discovery would likely escalate into a national security matter and a formal investigation rather than a routine patch.[20] The failure mode that should worry an evaluator is not the model caught on the bench and thrown away. It is the one whose compromise surfaces only after it has already been operating inside a classified environment.

Opacity is not the only pressure on training data. The supply of it is also finite. High-quality public text data has been described as a kind of fossil fuel for model development, a limited resource that could be exhausted soon, and researchers project that the stock of human-generated public text will be fully utilized sometime between 2026 and 2032 if current trends continue, or earlier if models are overtrained.[21] The relevance here is not economic. As clean public data grows scarce, the responses to that scarcity each work against visibility: developers turn to synthetic data, to hurriedly licensed archives, and to lower-quality sources further down

---

[17] National Security Agency et al., AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems, joint cybersecurity information sheet, May 22, 2025.
[18] National Security Agency et al., AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems, joint cybersecurity information sheet, May 22, 2025.
[19] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.
[20] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.
[21] Pablo Villalobos et al., "Will We Run Out of Data? Limits of LLM Scaling Based on Human-Generated Data" (arXiv:2211.04325). The "fossil fuel" framing and annual dataset-growth figures are drawn from Nikhil Mulani, An Evolving AI Supply Chain (University of California, Berkeley, October 2025), "Data: Scarcity and Cost."

the web. Synthetic data in particular has been shown to degrade model performance when relied on too heavily, and AI-generated content circulating back into training sets creates contamination feedback loops that are, by construction, undocumented.[22] Scarcity pushes the data supply toward exactly the conditions this section has been describing: less provenance, more unvetted material, a wider surface for the poisoning discussed above.

This scarcity does not press on everyone equally. As the public-data stock declines, the advantage shifts toward the largest firms, those already holding vast proprietary user data and positioned to benefit from a feedback loop in which more users generate more data, which trains better models, which attract more users still.[23] The strength of that loop varies: not all usage generates data that improves the model, and the empirical evidence on returns to additional training data is mixed. Hence, the advantage is real but not uniform across applications.[24] The concentration it points toward, however, is already visible at the foundation-model layer, where a single model, OpenAI's GPT-4, accounted for roughly 69 percent of generative-AI revenue in 2023 despite a crowded field of competitors.[25] For a program office, the practical consequence is that the training data behind a given model is not only hard to inspect but increasingly held by a shrinking set of firms with little obligation to disclose it.

Inherited provenance is only half of the concern. Once a program office or integrator takes possession of a model and fine-tunes it on proprietary data, it generates a modification history of its own, one it is in a position to document even when the model's origins are opaque. Whether that history is tracked is a choice. Chris Ritter, described an integrator model in which Dell vets a model before deployment and maintains its own record of the changes made and the data inputs used, rather than relying on the vendor to account for them.[26] The provenance a buyer cannot recover from the publisher is distinct from the provenance a buyer creates and can choose to keep.

For a government program office, this is the point at which data provenance stops being an abstract concern and becomes a procurement one. A model reviewed for a defense application arrives with a training history that cannot be inspected, cannot be reconstructed, and, in the cases above, was never fully disclosed by its publisher. The evaluator is asked to accept the

---

[22] On synthetic-data degradation, see S. Alemohammad et al., "Self-Consuming Generative Models Go MAD," arXiv:2307.01850, and Shiyin Ouyang et al., "Position: Sora Shows That Data Is More Important Than Compute," arXiv:2503.14023, both discussed in Nikhil Ragav Mulani, "An Evolving AI Supply Chain" (Berkeley). Mulani identifies data contamination feedback loops, in which public AI-generated content and private synthetic data pollute training datasets, as a forward-looking risk to data quality over time.
[23] Leonardo Gambacorta and Vatsala Shreeti, The AI Supply Chain, BIS Papers No. 154 (Bank for International Settlements, March 2025), "Training data" and "Foundation models" sections.
[24] On synthetic-data degradation, see S. Alemohammad et al., "Self-Consuming Generative Models Go MAD," arXiv:2307.01850, and Shiyin Ouyang et al., "Position: Sora Shows That Data Is More Important Than Compute," arXiv:2503.14023, both discussed in Nikhil Ragav Mulani, "An Evolving AI Supply Chain" (Berkeley). Mulani identifies data contamination feedback loops, in which public AI-generated content and private synthetic data pollute training datasets, as a forward-looking risk to data quality over time.
[25] Leonardo Gambacorta and Vatsala Shreeti, The AI Supply Chain, BIS Papers No. 154 (Bank for International Settlements, March 2025), "Training data" and "Foundation models" sections.
[26] Author interview with Chris Ritter, Chief Federal Application Architect, Dell Technologies (Federal), July 14, 2026. Ritter described Dell's role as an integrator that vets a model before deployment and maintains its own record of the modifications made and the data inputs used, rather than relying on the model's vendor to account for them.

model on the strength of a dataset name and a partial description. That is a structural gap, not negligence. Neither Florence-2 nor OWLv2 was designed with military supply chain governance in mind, and nothing about how they are published requires it. The visibility has to come from somewhere else, because it is not coming from the data. But where does that visibility come from?

Foundation Model Cascading Risk

Data provenance describes what you cannot see about a single model's training data. Cascading risk describes what happens when that invisible history is inherited. Most AI systems today are not trained from scratch. They are built on top of a pre-trained foundation model, which is itself often built on top of another. Each layer inherits everything beneath it: the training data, the architecture choices, the known and unknown weaknesses, and it inherits them automatically, often without the downstream developer knowing the dependency is there at all.

There are two distinct things a downstream developer fails to inherit visibly, and it is worth separating them, because they are different problems. The first is the data gap: the training history is embedded in the model's weights and cannot be read back out, so a developer who downloads a foundation model does not know what went into it. The second is the weight gap. Even setting the data aside, the weights themselves encode what the network actually does, and that encoding is not human-readable. As Ernst describes it, a modern foundation model has so many degrees of freedom that it becomes mathematically hard to understand what it is doing directly; the only practical approach is to test known inputs against expected outputs and infer the rest, with no way to verify every possible input. A downstream model inherits both gaps at once, and each layer added on top inherits them again.

Ernst frames the build process as a series of layers, and the framing is useful because it locates exactly where visibility is lost. The first layer is architecture, the model's blueprint. This layer is designed by research labs, large technology companies, academic institutions, and specialized firms, and it is the most transparent of the three: the structure is visible and inspectable, and very few developers build here at all. The second layer is foundation model training, where that architecture is trained on a large, generic, often internet-scale dataset. This is where provenance goes murky, as the full training history becomes invisibly embedded in the weights. The third layer is fine-tuning, where the foundation model is specialized on new domain data, and where, in Ernst's words, "you're layering things you don't know. You have no idea what went in there."[27] The resulting model carries both the foundation's unrecoverable history and the fine-tuning data's history on top of it. Fine-tuning is not a single event. It can happen repeatedly and at different hands along the chain: by the publisher, by an intermediate vendor, or by the end buyer training on its own proprietary data. Each pass is another layer of the kind Ernst describes. The distinction that matters for provenance is who performed it: the upstream layers are the ones a downstream user cannot see, while a buyer's own proprietary fine-tune is the one layer it is positioned to document.

[27] Author interview with Jan Ernst, Director of AI, Latent AI, June 23, 2026.

The models from the provenance section show the layering concretely. OWLv2 is built on CLIP: it takes the CLIP backbone and adds object detection heads on top, teaching it not just to recognize what is in an image but to locate and box specific things described in text. That is two layers of inherited dependency in a single model. Whatever provenance gaps exist in CLIP's roughly 400 million undocumented image-text pairs travel silently into OWLv2, and into everything built on OWLv2 in turn. Florence-2 shows the other direction of the same problem: its model tree lists dozens of fine-tuned and adapted versions, each a derivative with its own undocumented training history. A developer searching for Florence-2 could pull one of these believing it is the original. If someone had fine-tuned the model on adversarial or poisoned data and republished it under a slightly different name, nothing in the download process would flag the difference.

The joint guidance issued by the NSA, CISA, and the FBI in May 2025 states the principle directly. Inaccurate or compromised material, it warns, can compromise not only the models trained on that data but "any additional models that rely on compromised models as a foundation."[28] The compromise does not stay where it started. It propagates down the dependency chain by default.

This is a familiar failure mode in software, even if it is newer to AI. The clearest analog is Log4j: a critical flaw in an open-source logging library that a vast number of systems depended on, most without knowing they used it at all, because it arrived as a transitive dependency of something else. The vulnerability, later named Log4Shell, let attackers execute arbitrary code by getting the library to log a single crafted string — no authentication or user interaction required. When it was exploited in late 2021, the hard part for many organizations was not fixing it but discovering they had it. A compromised foundation model is the same problem moved up the stack: a flaw in a component nobody is looking at, inherited by everything on top.

There is one inheritance vector that goes beyond data and weights. Florence-2 requires trust_remote_code=True in every published code example, which means that using the model executes custom code shipped with it, code that runs on the developer's own machine without being audited first. A developer is inheriting not only an opaque training history but executable instructions. The paper returns to this in the trust_remote_code risk section; here it is enough to note that inheritance can carry code as well as history.

Not every expert reads the compounding the same way, and the disagreement is worth surfacing rather than smoothing over. Ernst's position is that each fine-tuning layer adds to the unknown history and therefore compounds the provenance problem. Kenyon offered a different view: because fine-tuning updates a model's weights, each round degrades what the model originally learned, so if poisoning were present in the original foundation model, successive fine-tuning might actually reduce the model's ability to act on it. Kenyon presented this as a hypothesis rather than an established finding.[29] In a later response, Ernst agreed it was plausible, for two reasons: fine-tuning narrows the problem domain and makes outputs easier to

[28] National Security Agency et al., AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems, joint cybersecurity information sheet, May 22, 2025.
[29] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.

validate empirically, and it may mathematically smooth the model's output distribution.[30] The two positions are not actually a contradiction. They measure different things: Kenyon is describing practical risk exposure, while Ernst is describing epistemic uncertainty about origins, and both can hold at once. The paper takes up this disagreement in full in its discussion of field disagreements.

For a government program office, the cascading structure changes what it even means to evaluate a model. The model in hand is the visible tip of a dependency chain reaching back through fine-tunes to foundations to datasets, and none of those layers is something the evaluator can inspect or the publisher is required to disclose. Accepting the model means accepting everything underneath it, sight unseen. The risk is not that any particular model is compromised. It is that the model's trustworthiness rests on layers no one in the acquisition chain can see. As with provenance, the visibility has to be built in from somewhere else, because the dependency chain will not supply it on its own.

Reproducibility and Verification

The previous sections describe what a developer cannot see. This one describes what a developer cannot prove, and the two are not the same. A model can arrive with a documented dependency list and still be impossible to reproduce. Verification is the harder standard: not "can I see what went in," but "can I independently rebuild what you built and confirm that the thing running in the field is the thing that was reviewed." For AI systems at the edge, the honest answer today is usually no.

Aronchick framed the goal precisely. The aim, he said, is a deterministic way to rebuild what was built, to be able to say unequivocally that the thing that ran came from a specific, verified set of source materials.[31] That is the standard software supply chain security has been moving toward for years, and it already has real, mature tooling behind it. A SBOM inventories describing a software artifact is made of. SLSA adds verifiable guarantees about how that artifact was built. The problem is not that these tools do not exist. It is where they stop.

They stop at the model boundary. As Aronchick put it, SBOM and SLSA were built for the binary and the pieces that surround the model, not for the model itself. They can attest to the container, the framework version, and the operating-system image, the software wrapper, but not to the thing inside the wrapper that determines the system's actual behavior. A model card cannot close this gap either. Aronchick was direct: a model card cannot reproduce a model. A well-formed binary manifest lists everything needed to rebuild the binary; a model card, at best, lists what a model was nominally built from, and does so as narrative rather than as a verifiable build specification. This is the same limitation of the provenance section found in the Florence-2 and OWLv2 model cards, seen from a different angle. There the cards were incomplete; here the point is that even a complete one is the wrong kind of artifact for verification.

[30] Written follow-up from Jan Ernst, Director of AI, Latent AI, July 7, 2026.
[31] Author interview with David Aronchick, CEO, Embedl, July 15, 2026.

Aronchick's bill-of-materials distinction organizes the whole problem, and it is worth stating plainly.

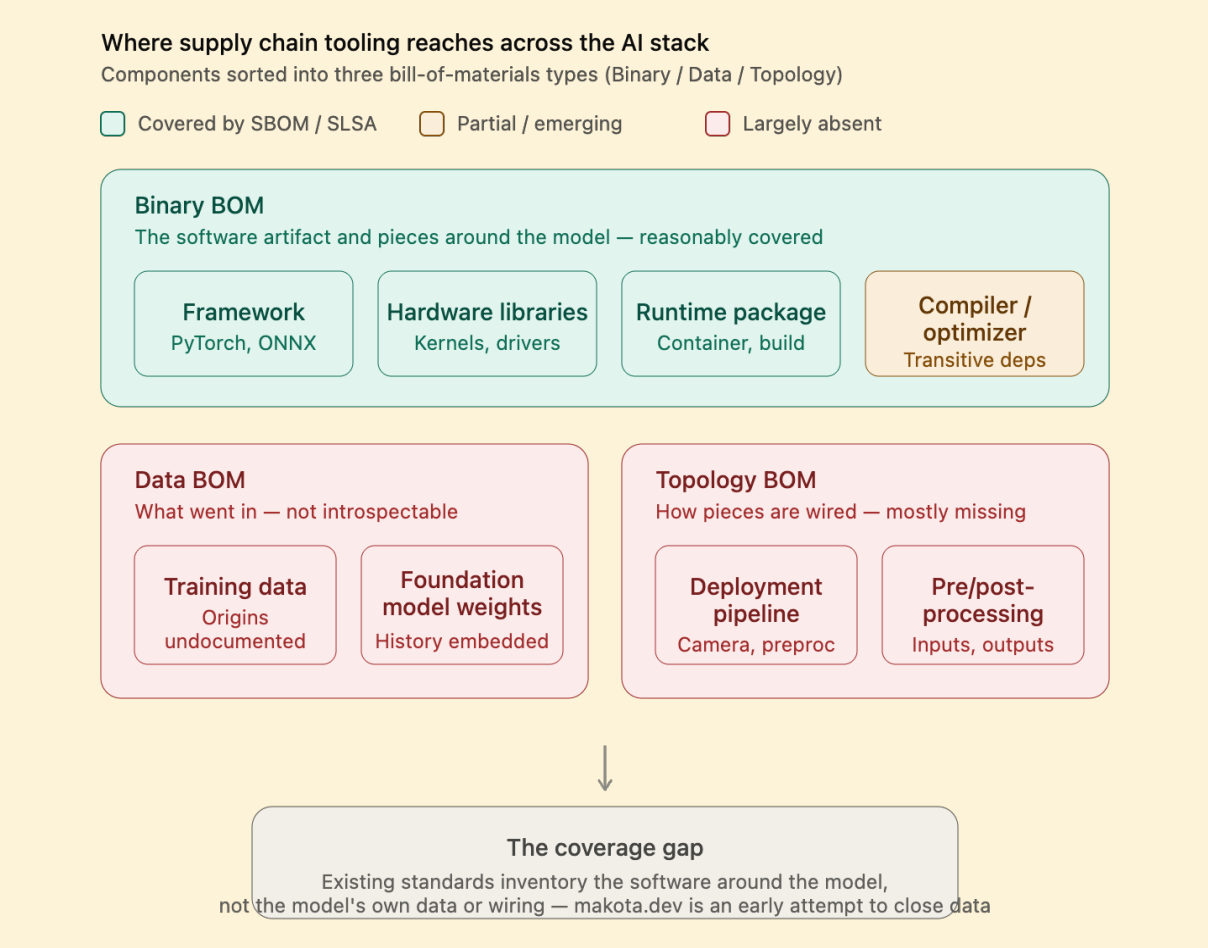

Figure 2. *Supply chain tooling coverage across the AI stack, with components sorted into Binary, Data, and Topology bills of materials. Framework after Aronchick; coverage assessments by the authors.*

There is a Binary BOM: the software artifact and its surrounding pieces, reasonably covered by existing tooling. There is a Data BOM: what the model was trained on, which Aronchick called an utter black box, not introspectable, and where he located the single largest gap, because provenance and lineage there sit completely untouched by current standards. And there is a Topology BOM: how the pieces are wired together and deployed, which he described as poorly handled today. Most of the component pieces of an AI system, in his assessment, do not have bills of materials at all right now.

Even within the binary layer, where coverage is supposedly strongest, the verification is shallower than it appears. Most languages have dependency-tracking mechanisms; Aronchick pointed to formats such as Go's go.sum and Python's pyproject.toml, configuration, which record which versions of which packages a build used. But a great deal is still missing. Instead of capturing a cryptographic hash that pins the exact artifact, these files often capture only a version number, which does not guarantee the bytes are identical. And transitive dependencies,

the dependencies of your dependencies, may not be captured at all. Aronchick was clear that this is not a new problem, but that AI is entering a world where it gets worse.

The practitioner interviews confirm the scale of what goes untracked. Abelardo Lopez, Director of Compiler and Embedded Systems at Latent AI, estimated that 50 to 70 percent of a final solution is based on open-source code, depending on the project.[32] Griffin put the open-source share of his container stack higher still.[33] Both independently identified transitive dependencies as the shared blind spot: direct dependencies are mostly visible, but what those dependencies pull in underneath is not fully tracked, and closing that gap takes deliberate effort rather than anything automatic. Neither described this as negligence. Supply chain provenance had simply never been a formal step in their development process, because the industry has not established one to adopt. And both, unprompted, reached for the same missing instrument: a proper bill-of-materials process to determine what is actually present. That two engineers working at different layers arrived at the same answer without prompting is itself a finding. The tooling they want does not yet reach where they need it.

There is also the question of what the available documentation actually certifies. Kenyon made the sharp observation that a Hugging Face model card represents the uploader's claims, not verified facts, and should be treated accordingly.[34] This closes the loop with the reproducibility problem. If a model card is an unverified claim rather than an attested build record, then the one document a downstream developer is handed is precisely the document that cannot be relied on for verification. The gap is not only that the standards stop at the model boundary; it is that what sits on the other side of that boundary is self-reported.

The edge context adds a dimension that data-center reproducibility does not have to reckon with. Aronchick raised it directly: the way a model is deployed is itself part of what must be reproducible. If a system reads from a camera at a particular angle, with particular preprocessing, that pipeline has to be reproducible too, and in edge deployments there are many different physical angles and configurations that each need to be captured for the system to keep improving. This is the Topology BOM problem made concrete, and it matters first for something basic: debugging. If you cannot reproduce the pipeline, you cannot reliably diagnose why a fielded model is failing, let alone certify that a replacement unit behaves like the one it replaced.

None of this is for lack of anyone having tried to standardize model metadata. The work exists; it simply predates the problem. The ML-Schema, developed within a W3C community group and first published in 2018, is a shared vocabulary for describing machine-learning algorithms, datasets, models, and experiments, built expressly to make experiments interoperable and reproducible across different platforms.[35] Its class structure reads almost like a first draft of what

[32] Author interview with Abelardo Lopez, Director and Compiler of Embedded Systems, Latent AI, June 17, 2026.
[33] Author interview with Mark Griffin, Director of Software Architecture, Latent AI, June 18, 2026.
[34] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.
[35] Gustavo Correa Publio et al., "ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies," ICML 2018 Workshop on Reproducibility in Machine Learning (arXiv:1807.05351). Developed within the W3C Machine Learning Schema Community Group; the effort is ongoing, so cite the 2018 release as its origin rather than its current state.

a model bill of materials would need to record: dataset and dataset characteristics, algorithm and implementation, hyperparameters, model and model characteristics, model evaluation, and the individual run that produced a result. In other words, the categories a Data BOM must capture have already been mapped. But the effort dates to the classical machine-learning era, and it addresses reproducing an experiment from its algorithm, dataset, and parameters, not verifying that a deployed set of weights is the one that was reviewed, nor tracing internet-scale training data, fine-tuning lineage, or supply-chain integrity. The vocabulary for describing a model exists. What does not exist is a widely adopted way to bind that description to a specific artifact and verify it at build time. Newer efforts are beginning to target exactly that binding problem on the data side; Aronchick pointed to makota.dev, which he described as inspired by SLSA but built for data, able to encode verification that a user holds all the source materials.[36]

For a government program office, this reframes the entire evaluation problem. The tooling that exists—SBOM, SLSA, and dependency manifests—can verify the wrapper around a model with reasonable confidence. It cannot verify the model, its training data, or the deployment pipeline it runs in. An evaluator who assumes an SBOM covers the AI system inherits a false sense of completeness: the bill of materials stops exactly where the model's behavior begins. As with provenance and cascading risk, verification has to be built deliberately into the places the standards do not yet reach, because the field, for now, has no standard that reaches them.

Stack / Pipeline Fragility

The risks described so far are about what a model is: its data, its inherited history, and whether it can be verified. This section is about what happens to a model in motion, as it is transformed for deployment and updated after deployment. Both are supply chain events, and both are points where a system can break or be broken without anyone having touched the model's weights directly.

The first fragility is in transformation. A model does not travel from repository to edge device unchanged. It passes through quantization, optimization, and compilation, and each of those steps has its own toolchain with its own dependencies. As Lopez noted, the optimization tooling itself, the compilers and linkers that adapt a model for edge hardware, carries dependency chains of its own. The model that clears review and the model that actually runs on the device are not the same artifact; a chain of transformations sits between them, and that chain is rarely tracked with the same rigor as the model or its training data.

The build and deployment pipeline is also an attack surface in its own right, independent of the model moving through it. Griffin identified the CI/CD toolchain as a distinct risk layer: the pipeline can introduce CVEs or licensing obligations that travel with the artifact into deployment, and provenance across the full dependency tree is not automated, but takes deliberate effort. The concern is not hypothetical. The clearest precedent is the SolarWinds compromise. Attackers gained access to the company's network in 2019. Still, the consequential step came later: malicious code was inserted into the software build pipeline for its widely used Orion IT-management product, and trojanized updates were distributed to customers between March and

[36] Author interview with David Aronchick, CEO, Embedl, July 15, 2026.

June 2020. Up to roughly 18,000 organizations, including US government agencies, installed a compromised update. However, a much smaller subset was actively exploited, and the compromised updates circulated for months before the intrusion was discovered in December 2020.[37] The lesson SolarWinds carries into the AI context is that a fully legitimate, correctly signed model can still arrive compromised if the pipeline that packaged it was compromised. Verifying the model is not the same as verifying the path it traveled.

The second fragility appears after deployment, and it is the one Ritter spoke to most directly. A deployed model needs regular updates to patch vulnerabilities, correct for data drift, and adapt to changing conditions. In a data center, updating is trivial; under the same SWaP constraints discussed earlier, it remains one of the hardest problems in the system, and it is largely absent from existing supply chain discussions. At the edge, in a contested or disconnected environment, it is one of the hardest problems in the system, and it is largely absent from existing supply chain discussions.

Ritter described the deployment as a hub-and-spoke model: a hub, the Pentagon in his example, communicates with spokes that are the force elements operating at the edge, a battalion or a division running models locally. The difficulty is not simply maintaining a connection. As Ritter put it, the problem is no longer that systems like Starlink cannot reach anywhere in the world; the problem is electronic warfare. A device that broadcasts for any sustained period raises a flag, because an adversary does not need to read the traffic to act on it: the transmission itself becomes a location. Connectivity, in other words, is not just unreliable at the edge; using it can be dangerous.

That constraint reshapes how updates have to work. Ritter's approach is to push updates as very small, blueprint-like packages, kilobyte-scale files that update only the specific pieces that changed, reducing reliance on broadband and on connections that stay open long enough to be traced. He drew the analogy to how enterprise resource planning systems such as SAP push targeted updates to distributed sites. The same lightweight mechanism carries sensor and logistics data back the other way. The broader principle he stressed is that this software fabric- the data management, the model-update transport, and the mechanism for pushing changes to specific consumers- has to exist before deployment begins, not be improvised afterward. Ritter noted that his systems share model outputs rather than training data across networks, because moving raw data of that volume between distributed locations is untenable, a constraint that itself shapes what the pipeline can carry.

The failure mode this exposes is quiet. A model that cannot receive updates does not stop working; it degrades. As the operational environment shifts away from the conditions the model was trained on, data drift sets in, and the model's accuracy erodes silently. At the same time, it continues to report confident outputs. An edge system in a communications-denied environment

[37] SolarWinds Corp., Form 10-K (FY2020) and SEC disclosures; MITRE ATT&CK Campaign C0024 (SolarWinds Compromise). Malicious "SUNBURST" code was injected into Orion platform builds released between March and June 2020; SolarWinds reported that fewer than 18,000 of roughly 33,000 Orion customers may have used the compromised version, with follow-on exploitation confined to a much smaller subset.

is therefore exposed to a slow supply chain failure with no alarm attached: the update chain is severed, and the only symptom is a model quietly becoming wrong.

Underneath both fragilities sits a chain-of-custody question that current procurement does not resolve. For any update reaching a fielded device: who approved it, who built it, who delivered it, and who verified it was not altered in transit? The May 2025 NSA, CISA, and FBI guidance speaks to this directly, treating every point at which new data or feedback enters a model as requiring the same verification and validation as the original training.[38] An update is not a maintenance action; it is a new supply chain event, and at the edge it may have to be trusted without the connectivity to verify it fully.

For a government program office, stack and pipeline fragility is where the supply chain stops being a static inventory and becomes a lifecycle. A model can be verified at the moment of acquisition and still be transformed by a toolchain no one inventoried, delivered through a pipeline no one attested, and left to drift in the field because the update mechanism was never designed for a contested environment. Visibility, here, is not a one-time check at procurement; it is a property the system either maintains across its whole deployed life or loses the moment the first update fails to arrive.

Permissive Licensing & Deployment Gap

Every risk to this point has been something a developer could, in principle, measure or trace. This one is different in kind. There is nothing hidden here, nothing that better tooling would reveal. The license on a model is public, plain, and permissive by design, and that is precisely the problem. The gap is not between what a developer can see and what they cannot. It is between what a model was licensed and intended for and what it is actually being used to do. No instrument detects that gap, because it is not a technical property of the artifact. It is a governance question wearing a technical disguise.

Start with the licenses themselves. Florence-2-large is released under the MIT license; OWLv2 under Apache 2.0.[39] Both are among the most permissive open-source licenses in wide use. As Lopez described them from practice, these are licenses that let you do essentially whatever you want with the software; Apache's only real obligation is attribution, a requirement to acknowledge that you used it, not a restriction on how.[40] He contrasted this with the license developers deliberately avoid, the GPL, which in his description forces any derivative work to be made public and so blocks commercialization. The working preference in the field runs toward

[38] National Security Agency et al., *AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems*, joint cybersecurity information sheet, May 22, 2025 (verification and validation required each time new data or user feedback is introduced).
[39] Model cards, Hugging Face: microsoft/Florence-2-large (MIT license) and google/owlv2-base-patch16 (Apache 2.0). Both licenses are confirmed in the models' repository metadata.
[40] Author interview with Abelardo Lopez, Director of Compiler and Embedded Systems, Latent AI, June 17, 2026. (Lopez characterized the GPL as forcing derivative work into the public domain; more precisely, the GPL is a copyleft license requiring derivatives to be released under the GPL. The practical effect he described, that it impedes commercialization, is the reason his team avoids it.)

exactly the permissive licenses these two models carry: maximum freedom, minimum obligation.

For most software, that permissiveness is a feature. For a model that may end up in a defense system, it means the license imposes none of the controls a defense supply chain would otherwise require. There is no chain-of-custody requirement. There is no obligation to notify the original publisher if the model is modified. There is no requirement to disclose modifications to downstream users. And there is no mechanism to track where the model goes after it is downloaded. A model can be taken, altered, rebundled, and passed down a commercial chain to a government buyer, and nothing in the license creates a record of any step in that journey.

The scale at which this happens is visible on the model's own page. Florence-2-large has been downloaded on the order of hundreds of thousands of times in a single recent month, with no verification of who is downloading it or for what purpose.[41] That figure is not a risk in itself; it is a measure of how thoroughly the question of downstream use has been left unanswered. Every one of those downloads is a point at which a capable computer-vision model enters a system its publisher has no visibility into and no license basis to ask about.

This is where the second half of the gap opens. The models state their intended audience plainly. The OWLv2 model card specifies that its primary intended users are AI researchers, and imagines the model being used to understand better the capabilities, biases, and constraints of computer-vision models.[42] It was not designed or validated for operational deployment, let alone military deployment. Yet the intended-use statement is exactly that, a statement. It has no enforcement mechanism, and the permissive license actively guarantees it cannot acquire one. Nothing prevents a developer from pulling either model into a fielded edge system today.

That this is already happening is not purely a matter of inference. Among the applications posted publicly on Hugging Face is one named, plainly, "weapon-detection-app," a publicly posted weapon-detection application, built on a research-oriented object-detection model, trained on data its publisher never fully documented, with no defense-grade supply-chain auditing required or performed at any point.[43] The intended-use line and the actual deployment are separated by the entire distance between a research disclaimer and a weapons application, and nothing in the model's licensing, documentation, or distribution did anything to hold that line.

It is worth being precise about what this does and does not indict. This is not a failure by Microsoft or Google. Both companies published capable models, documented them to the prevailing norms of open research, and licensed them the way open-source models are conventionally licensed. Neither was designed with military supply-chain governance in mind, and nothing about the way either was released required it to be. That is the point. The gap is structural, not negligent. It exists in the space between how open models are published and how they are actually being used, and it persists precisely because no one in that space did anything wrong by the standards they were operating under.

---

[41] Florence-2-large model page, Hugging Face.
[42] OWLv2 model card, Hugging Face (google/owlv2-large-patch14-finetuned), "Primary intended uses" section: "The primary intended users of these models are AI researchers."
[43] "weapon-detection-app," Hugging Face Space (user: marcuscanhaco).

For a government program office, the deployment gap reframes what "approved for use" can even mean. A permissive license is not a clearance. A research-use statement is not a control. When a model arrives, possibly several commercial hands removed from its origin, as later sections describe, its license history will confirm only that everyone along the way was permitted to do what they did, and will certify nothing about whether the model was ever meant for the use it has been put to. Visibility, here, cannot come from the license, because the license was written to impose as little as possible. It has to be supplied by whoever is accountable for the decision to deploy, and at present it is often not clear who that is.

Geopolitical Risk (the pressure)

The licensing section ended with a question: when a model arrives in several commercial hands removed from its origin, who is accountable for what it is and where it came from? Geopolitical risk is that question asked at national scale. It is also where this paper has to be careful about its own boundaries.

As Shanahan put it, the largest geopolitical exposures- export controls, sanctions, and rare-earth minerals- are real. Still, they operate at a level of complexity above software supply chain risk specifically, and they are already being managed at a high geopolitical level.[44] Those belong to the hardware and capital story told in the Berkeley report this paper refers to.[45] The concentration of advanced chip manufacturing and critical-mineral processing has already drawn export controls and national industrial policy in a way the software layer has not, a hardware dependency this paper flags but, as noted in the anatomy section, leaves to others to trace.[2] What remains, and what is squarely in scope, are the parts of the geopolitical problem that travel through software: differing data-governance regimes, economic displacement, and a market dynamic that turns cost pressure into strategic dependency.

The first in-scope piece is that data governance is not uniform across borders. A model, its training data, and its retraining pipelines may pass through jurisdictions with fundamentally different rules about who may access data and under what authority, the divergence between, for instance, European privacy law and state data-access regimes elsewhere. For a model whose provenance is already opaque, a supply chain that crosses those regimes adds a second layer of uncertainty: not only is it unclear what data a model saw, but it can be unclear which legal regime governed that data when it was collected, or where it is processed on update. Shanahan confirmed this as a legitimate, in-scope concern, naming differing data-governance regimes directly.[46]

---

[44] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026. The "70 percent of US startups" figure is Shanahan's own recollection, which he hedged in the interview ("if I have the number right"); it is presented here as his estimate, not an established statistic.
[45] Nikhil Mulani, *An Evolving AI Supply Chain* (University of California, Berkeley, October 2025), "Hardware: Specialization and Geopolitics." The hardware, compute, and capital layers are mapped there at length; this paper places the hardware dependency without tracing it.
[46] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026. The "70 percent of US startups" figure is Shanahan's own recollection, which he hedged in the interview ("if I have the number right"); it is presented here as his estimate, not an established statistic.

The second, and the one Shanahan singled out as likely to surprise most readers, is economic displacement operating as a capture strategy. By his account, roughly 70 percent of US startup companies are building on Chinese open-source models, DeepSeek v4 in particular, for a straightforward reason: cost.[47] For a young company, the deciding factor is rarely capability at the margin; it is runway. When the cheapest capable option is a Chinese open-source model, and with major US labs having largely moved to closed source, Shanahan argued it often is, a cash-constrained startup will take it. He framed the dynamic in blunt terms: undercut on cost or terms of access, become embedded, and raise the cost of leaving once dependence has set in. In his phrasing, the goal is to "get sticky," to reach a position where a model is "just too hard to pull out," and then to convert that entrenchment into leverage over time. He likened it explicitly to a land-and-expand commercial playbook.

What makes this a supply chain problem rather than a market-share story is what happens next: the dependency doesn't stay contained to the startup that made the choice. It propagates to every customer, integrator, and downstream system that builds on that startup's product — the same structural gap the licensing section described, now running through a chain of companies. A startup builds on DeepSeek v4. It sells a finished package to another US company, which integrates it and sells a complete solution to the government, or the startup sells to the government directly. At no point in that chain does the underlying dependency on a foreign open-source model necessarily become visible. The government buyer sees a US vendor and a finished capability; it does not see, and has no automatic way to see, that the foundation underneath was built on a model it would never have knowingly procured. Shanahan connected this directly to the paper's cascading-risk argument: the dependency is inherited silently, and the visibility to detect it does not exist at any handoff.

He acknowledged a reflexive wariness toward anything built on a PRC open-source model, while being equally clear that this was an instinct rather than a finding, and that what he wanted was an analysis, not an assumption. As he put it, the situation calls for “a realistic assessment of tell me what the risks are out there.” The honest position is narrower than the alarm: the concern is not that DeepSeek-based models are known to be compromised, but that the dependency is invisible, the leverage is real, and no one in the procurement chain is currently positioned to know the dependency is even there. The risk is blindness, not a proven payload. The honest position is narrower than the alarm: the concern is not that DeepSeek-based models are known to be compromised, but that the dependency is invisible, the leverage is real, and no one in the procurement chain is currently positioned to know the dependency is even there. The risk is blindness, not a proven payload.

For a government program office, geopolitical risk in this narrower, software sense collapses back into the paper's central problem. The question is not whether to fear foreign models in the abstract; it is whether the acquisition process can see a foreign dependency when one is present. At the moment, across a chain of commercial vendors each selling to the next, it

[47] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026. The "70 percent of US startups" figure is Shanahan's own recollection, which he hedged in the interview ("if I have the number right"); it is presented here as his estimate, not an established statistic.

frequently cannot. The strategic exposure and the visibility gap are the same gap. Closing the second is the only reliable way to manage the first.

Stakeholder Accountability

Every previous section ended at the same place: the visibility must come from somewhere, and it is not coming from the artifact, the license, or the supply chain on its own. This section is about where it comes from, and its central finding is that, at present, no one is clearly responsible for supplying it. The risks this paper describes are not going ungoverned because anyone decided they should be. They are ungoverned because the rules that would assign responsibility do not yet exist. Developers are not being negligent. They are operating without a map.

That phrase is not rhetorical; it describes a specific policy condition, laid out most directly by Morgan Plummer, a defense insider who served as managing director of the DoD's National Security Innovation Network and as a professor of practice at the US Air Force Academy, in a March 2026 op-ed.[48] Plummer argues that the United States is governing military AI through a policy vacuum. Rather than statutory guardrails set by Congress, US policy relies largely on general Pentagon guidance calling for "appropriate levels of human judgment" in military systems, language broad enough to leave the operative questions unanswered.[49] When commercially developed AI is adapted for national security use, particularly for applications involving lethal force, and no clear rules govern that adaptation, the boundaries end up negotiated in real time, case by case, by the people building and fielding the systems.

The reason this vacuum matters now, rather than in the abstract, is that the identity of who builds military AI has changed. Plummer's observation is that advanced AI, software, and autonomous systems are increasingly built not inside traditional defense contractors but in startups, research labs, and technology firms whose primary markets are commercial, organizations with little history working with the defense establishment and little reason to have internalized its supply-chain governance norms.[50] The builders and the governance framework have drifted apart: the companies now producing the capability were not shaped by defense acquisition culture, and defense acquisition rules were not written for them.

The Anthropic dispute is that vacuum made concrete. In July 2025, the DoD's Chief Digital and Artificial Intelligence Office awarded Anthropic a two-year prototype agreement with a $200 million ceiling to deploy its Claude models on classified government networks, where Claude became the first frontier model approved for use on such networks.[3] The relationship then broke

[48] Morgan Plummer, “A Policy Gap Is Threatening the Pentagon’s AI Innovation Pipeline,” *DefenseScoop*, March 2026. Plummer is Vice President of Policy at Americans for Responsible Innovation; he previously served as Managing Director of the DoD National Security Innovation Network and as a Professor of Practice at the U.S. Air Force Academy.
[49] The phrase "appropriate levels of human judgment" is from DoD Directive 3000.09, *Autonomy in Weapon Systems* (updated January 2023), quoted here via Plummer's critique of its breadth.
[50] Morgan Plummer, “A Policy Gap Is Threatening the Pentagon’s AI Innovation Pipeline,” *DefenseScoop*, March 2026. Plummer is Vice President of Policy at Americans for Responsible Innovation; he previously served as Managing Director of the DoD National Security Innovation Network and as a Professor of Practice at the U.S. Air Force Academy.

down over the terms of use. By Anthropic's account, as the parties negotiated deployment on the department's GenAI.mil platform, the department sought terms permitting "all lawful uses," and Anthropic declined to remove two restrictions it maintained: that its models not be used for mass domestic surveillance or for fully autonomous weapons.[51] In late February 2026, the administration directed federal agencies to cease using Anthropic's technology. In early March 2026, through letters dated March 3, the Department formally designated Anthropic a supply chain risk to national security, reportedly the first time an American company had received that designation.[52] Anthropic challenged the designation in federal court on March 9, 2026, filing in two jurisdictions; a district court granted a preliminary injunction later that month, and the government appealed.[53] The dispute remained unresolved as this paper was written.

Whatever its eventual legal resolution, the dispute illustrates the governance gap precisely, and independently of who was right. This paper takes no position on the merits; it cites the episode as a governance case study, not as a claim about either party. The core problem it exposes is that a capability was deployed onto mission-critical national security networks before the terms governing its use, what it could be used for, what restrictions traveled with it, and who controlled its updates were settled. Those questions surfaced as a crisis after deployment rather than as conditions before it. That is what a policy vacuum produces: not necessarily a bad actor, but a set of foundational questions that get answered only when a conflict forces them, at the worst possible moment to be answering them.

There is a structural reason the vacuum persists, and Ritter named it. The problem, in his account, is that people approach these systems as a vertical output, each organization solving for the piece it owns, inside its own boundary of authority. But an edge AI capability in a contested environment is a horizontal problem: it touches every piece of hardware, every communications component, every policy and procedure, and every link in the supply chain from manufacturer to field. No one operating within a single vertical owns that horizontal reality. And when everyone integrates their own piece under their own rules, their own authentication methods, their own access models, their own underlying technology, the seams between those pieces are where security breaks down, and gaps appear.[54] Ritter's point is that closing the vacuum is not only a matter of writing rules; it requires a single authority positioned above the verticals to own the horizontal whole, and that role frequently does not exist in a vertically organized institution.

---

[51] Anthropic–DoD prototype agreement (July 2025), reported at a $200 million ceiling; deployment terms and the two use restrictions (mass domestic surveillance; fully autonomous weapons) per Anthropic's account and contemporaneous reporting. See coverage in the Center for American Progress analysis and legal-industry summaries (e.g., Mayer Brown, Goodwin, Wiley) of the dispute.

[52] Supply-chain-risk designation notified by letters dated March 3, 2026; described in reporting as the first such designation applied to an American company. Preceded by a late-February 2026 directive to federal agencies to cease use.

[53] Anthropic filed suit on March 9, 2026 (Northern District of California and the D.C. Circuit); a district court granted a preliminary injunction on March 26, and the government appealed to the Ninth Circuit on April 2nd.

[54] Author interview with Chris Ritter, Chief Federal Application Architect, Dell Technologies (Federal), July 14, 2026.

Shanahan reached the same conclusion from the governance side. He identified the policy vacuum as one of the most important risks on the list, notable given that his own current work sits squarely in this area.[55] His definition of governance is a useful description of exactly what the vacuum is missing: policy on which models may be used and for what, disclosure requirements, accountability when a model causes harm, and lifecycle management specifying who is permitted to update a fielded model and who verifies those updates.[56] Disclosure is the part of that framework most directly at stake here. A vendor may have done the provenance work internally and still be under no standing requirement to share it; when a program office accepts the vendor's assurances in place of the underlying record, the gap is not that the work was never done but that no one is positioned to require its transfer. He was candid that the line is blurry, observing that "it's kind of a little bit hard to say where supply chain risk stops and these things begin. They're all tied together."[57] That entanglement is the point. The supply-chain visibility problem this paper traces and the governance vacuum are not separate issues; the first cannot be managed without the second, because visibility with no one accountable for acting on it changes nothing.

For a government program office, the policy vacuum is the risk that contains all the others. Data provenance, cascading dependency, unverifiable builds, fragile update pipelines, permissive licensing, silent foreign dependency: each is serious on its own, but each is ultimately a question of who is responsible for seeing it and empowered to act. Right now, across a chain that runs from an undocumented dataset through a foundation model, a fine-tune, a commercial integrator, and a prime contractor to a fielded edge device, that responsibility is diffused to the point of absence. The visibility this paper argues for is not, in the end, a technical artifact. It is an accountable owner, someone positioned to demand the provenance, verify the build, control the updates, and carry the risk decision knowingly rather than inherit it blind. Building that role is the work the vacuum leaves undone.

[55] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.
[56] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.
[57] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.

6. Case Studies in Practice

The two models this paper has returned to throughout are not hypotheticals and not obscure. Florence-2-large and OWLv2 are capable computer-vision models published by Microsoft and Google, each a plausible candidate for edge deployment, each downloadable today by anyone, with no verification of who is taking them or why. The preceding sections examined them one risk at a time. The purpose here is to set them side by side and see the whole picture at once, because the risks this paper describes do not arrive one at a time. They arrive together, on the same artifact.

Risk area	***Florence-2-large (Microsoft)***	***OWLv2 (Google)***
Data Provenance	*Model card names the training set (FLD-5B: 5.4 billion annotations across 126 million images) but does not document where the images came from, who collected them, who owns them, or what they contain.*	*Model card names some sources (YFCC100M, COCO, OpenImages) but attributes a large share of the data to internet crawling without documenting that portion.*
Foundation Model Cascading Risk	*Trained by Microsoft on FLD-5B; the undocumented origins sit in that dataset rather than in a separately inherited backbone.*	*Built on the CLIP backbone, so it inherits the undocumented origins of roughly 400 million scraped image-text pairs it never saw directly. Two layers of inherited dependency.*
Reproducibility and Verification	*Cannot be independently rebuilt or verified against Microsoft's claims; the model card records a nominal history, not a verifiable build specification.*	*Cannot be independently rebuilt or verified against Google's claims; the same limitation applies through the CLIP layer beneath it.*
Stack / Pipeline Fragility	*Once compressed and containerized for edge hardware, the deployed artifact diverges from the reviewed model; transitive dependencies of the runtime remain a blind spot.*	*Same transformation exposure at deployment; the smaller parameter count makes it a stronger true-edge candidate, so the compression-and-containerization step is more likely to be exercised.*
Permissive Licensing & Deployment Gap	*Released under the MIT license: permits military use while requiring no chain of custody, notification, or tracking in return.*	*Released under Apache 2.0: also permits military use with nothing required in return. Model card names AI researchers as intended users, yet a weapon-detection application has already been posted publicly on the same repository.*
Geopolitical Risk	*Published by a US company; the concern is downstream, where the model and any retraining pipeline may pass through jurisdictions with differing data governance.*	*Published by a US company; same downstream concern, compounded by the multi-layer dependency inherited through CLIP.*
Stakeholder Accountability	*Downloadable by anyone with no verification of who is taking it or why; no document establishes who is accountable once it enters a defense system several commercial hands removed from Microsoft.*	*Same open distribution and the same unanswered question of accountability once it enters a defense system several commercial hands removed from Google.*

Table 1. *Florence-2-large and OWLv2 across the seven core risk areas. Conditions as described in Sections 5 and 6, drawn from the models' Hugging Face model cards and licenses; figures to be re-pulled at publication.*

Take OWLv2 alone and follow what a developer actually inherits by pulling it. Its own training data is disclosed only in part: the model card names some sources, such as YFCC100M, COCO, and OpenImages, but attributes a large share of the data to crawling the internet without documenting what that portion contains, which is the provenance gap.[58] It is built on top of CLIP, so it silently inherits the undocumented origins of roughly 400 million scraped image-text pairs it never saw directly, which is the cascading dependency. Nothing about it can be independently rebuilt or verified against what its publisher claims, which is the reproducibility gap. Its Apache 2.0 license permits military use while requiring nothing in return, no chain of custody, no notification, no tracking, which is the licensing gap.[59] Its model card names AI researchers as its primary intended users, yet a weapon-detection application has already been posted publicly on the same repository, which is the deployment gap.[60] And beneath all of that sits the question no document answers: who is accountable for any of it once it enters a defense system several commercial hands removed from Google, which is the policy vacuum.

Every risk this paper has traced across the preceding sections converges here, on one model of roughly 200 million parameters, downloadable in a single line of code, with none of these questions asked or answered at any point along the way. That convergence, not any single risk in isolation, is the condition this paper is written to make visible. The visibility to see even one of these gaps does not arrive with the model; the visibility to see all of them at once has to be built deliberately, because nothing in the artifact, the license, or the distribution supplies it.

[58] OWLv2 model card, Hugging Face (google/owlv2-base-patch16): training-data description and Apache 2.0 license.
[59] OWLv2 model card, Hugging Face (google/owlv2-base-patch16): training-data description and Apache 2.0 license.
[60] "weapon-detection-app," Hugging Face Space (user: marcuscanhaco).

7. Where the Field Still Disagrees

The risk sections presented a settled picture because, on the core facts, the experts agree: provenance is opaque, dependencies are inherited, and verification tooling stops at the model boundary. But on two questions the people closest to this problem genuinely disagree, and the disagreements are not noise to be resolved before publishing. They mark the edges of what is actually known.

The first is whether fine-tuning compounds inherited risk or erodes it. The second is where the risk primarily lives: in the model itself, which is the frame this paper has used throughout, or in the environment and tooling built around it. Each disagreement is set out below in the participants' own terms, with their hedges intact, because the honest position is to show where the evidence runs out rather than to paper over it.

Fine-tuning and compounding risk

The cascading-risk section presented fine-tuning as an accumulation of hidden history: each layer, in Ernst's framing, adds another round of training data whose origins cannot be traced, so a deployed model may carry three or more strata of unknown provenance.[61] On that view, fine-tuning makes the visibility problem strictly worse.

Kenyon offered a mechanism that points the other way. Because fine-tuning updates a model's weights, he noted, each round potentially degrades what the model had originally learned, pulling it progressively away from its initial training.[62] He extended this to poisoning as a specific case: if a compromise were baked into the original foundation model, then each subsequent fine-tuning step, by eroding the original learning, might also erode the model's ability to act on that compromise. He was careful to frame this as conjecture rather than a substantiated claim. And he was precise about what he was actually claiming: that fine-tuning degrades the foundation model's ability to do what it was originally trained to do, rather than necessarily compounding the poisoning itself.

Presented with Kenyon's mechanism, Ernst did not treat it as a rival. In a written follow-up, he agreed it was plausible and offered two reasons.[63] First, fine-tuning narrows a model to a specific problem domain, which makes its outputs easier to validate empirically; you can test what it now does against the cases you care about, regardless of what it once learned. Second, and more speculatively, fine-tuning may mathematically smooth a model's output distribution, softening sharp learned behaviors. He flagged the second reason as conjecture, in the same spirit as Kenyon's own hedge.

[61] Author interview with Jan Ernst, Director of AI, Latent AI, June 23, 2026.
[62] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.
[63] Written follow-up from Jan Ernst, Director of AI, Latent AI, July 7, 2026.

The two positions look contradictory only if they are answers to the same question. They are not. Ernst's original claim is about epistemic uncertainty: with each layer, you know less about what went into the model. Kenyon's claim is about practical risk exposure: with each layer, a specific latent behavior may become less operative. Both can be true at once. A heavily fine-tuned model can be at the same time more opaque about its origins and less likely to express any particular thing buried in those origins. Understood this way, the disagreement is not a conflict to adjudicate but two instruments reading two different dimensions of the same event, and the honest takeaway is that the field does not yet have the evidence to say how those dimensions net out for any given model.

Where Does Visibility Risk Actually Live?

The second disagreement is more fundamental, because it challenges this paper's organizing premise rather than a detail within it. This paper is built on the argument that visibility into a model's supply chain, its data, its lineage, and its modifications, is the thing worth fighting for. Kenyon, who agreed readily that this visibility is absent, does not fully accept that it is the priority.

His position, stated plainly, is that visibility into the model's supply chain by itself does not matter as much as it might seem. What matters more, in his assessment, is the software environment around the model: the tools, the engineering scaffolding, and the vulnerabilities living there.[64] He located the more urgent supply chain risk not in the model but in the systems that wrap it, singling out agent-based systems, which he characterized as "even worse from a supply chain risk standpoint because it's kind of just wild west."[65] The reasoning connects to his fine-tuning point. If a model is fine-tuned, optimized, and quantized until it is, in effect, no longer the same model as the foundation it started from, then interrogating the lineage of the original foundation model may be less valuable than evaluating the actual artifact in its actual environment, empirically, for the use case at hand.

This deserves to be taken seriously rather than deflected, because Kenyon is not wrong about anything factual. Model-level provenance genuinely does not detect a poisoned dataset; his own point elsewhere, that an SBOM does nothing to surface data poisoning, cuts exactly this way. Evaluation against a representative dataset genuinely is, as he argued, the only reliable way to catch a compromise. And the tooling around models, agent frameworks especially, genuinely is less mature and less scrutinized than the models themselves.

The paper's response is not to dispute any of that but to reframe the relationship as complementary rather than either-or. Kenyon is describing detection: how you catch a problem in the artifact you actually hold, right now, through evaluation and by securing its environment. This paper is largely describing accountability and prevention: how you know what you accepted, who is responsible for it, and what decision was made knowingly versus inherited blind. Evaluation tells you whether the model in front of you misbehaves today; provenance tells

[64] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.
[65] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.

you whether you should have trusted its path in the first place, and who answers for it if it fails tomorrow. A program office needs both, and they fail in different ways. A model can pass every evaluation and still carry an undocumented dependency that becomes a liability the moment conditions shift, exactly as the pipeline-fragility section described with silent data drift. Kenyon's challenge does not dissolve the paper's frame. It bounds it, a reminder that visibility is necessary but not sufficient, and that a visibility regime ignoring the surrounding software environment would solve only part of the problem.

That the strongest challenge to this paper's premise comes from one of its own interviewees is worth stating rather than hiding. It is also, in its way, the paper's argument in miniature: the honest position is not that visibility solves everything, but that its current near-total absence forecloses conversations the field has not yet had the chance to have.

8. Momentum to Address Supply Chain Risks

The problems described so far are not going unaddressed. Several efforts are underway to bring supply chain visibility to AI systems, and others, built for adjacent problems, are being looked to as models for what a solution might resemble. None is complete, and most were built for problems near the one this paper describes rather than for it directly. This section describes what exists and where each effort currently stops. It does not recommend one approach over another. That judgment belongs to the organizations that have to weigh cost, maturity, and mission fit for themselves.

SBOM for AI

The most developed effort is the extension of the software bill of materials to AI systems. In June 2025, the G7 nations set out a shared vision for applying the SBOM to AI, and in May 2026, the G7 Cybersecurity Working Group published a minimum-elements standard building on it.[66] The standard organizes an AI system's components into seven clusters: metadata, models, dataset properties, system-level properties, key performance indicators, security properties, and infrastructure.[67] The dataset-properties cluster is the one most relevant to this paper's concerns. It is meant to capture the provenance, content, sensitivity, statistical properties, and licensing of the datasets used across a model's lifecycle.[68]

On paper, this addresses much of the Data BOM gap described in the anatomy section. In practice, two limitations matter. First, an SBOM records what a producer declares; it does not verify those declarations, which returns to the problem Kenyon raised about model cards, where an inventory is only as reliable as the party filling it in. Second, and more fundamental, an SBOM does not detect data poisoning. Kenyon was direct on this point. Generating an SBOM tells you which components are present, not whether any of them were manipulated. The only reliable way to detect poisoning, in his account, is to evaluate the model against a diverse, representative dataset for the specific use case.[69] He dismissed the claim that a mathematical

---

[66] G7 Cybersecurity Working Group, *Software Bill of Materials (SBOM) for Artificial Intelligence: Minimum Elements* (May 12, 2026), building on the G7 "Shared Vision on SBOM for AI" (June 2025). Published with CISA and G7 partners; the seven clusters and the dataset-properties contents are drawn from that document. The minimum elements are voluntary recommendations, not binding requirements.
[67] G7 Cybersecurity Working Group, *Software Bill of Materials (SBOM) for Artificial Intelligence: Minimum Elements* (May 12, 2026), building on the G7 "Shared Vision on SBOM for AI" (June 2025). Published with CISA and G7 partners; the seven clusters and the dataset-properties contents are drawn from that document. The minimum elements are voluntary recommendations, not binding requirements.
[68] G7 Cybersecurity Working Group, *Software Bill of Materials (SBOM) for Artificial Intelligence: Minimum Elements* (May 12, 2026), building on the G7 "Shared Vision on SBOM for AI" (June 2025). Published with CISA and G7 partners; the seven clusters and the dataset-properties contents are drawn from that document. The minimum elements are voluntary recommendations, not binding requirements.
[69] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.

shortcut could substitute for that evaluation, calling it "snake oil."[70] An SBOM is a necessary inventory. It is not, by itself, a security control.

The limits are not only Kenyon's assessment. Allan Friedman, who led CISA's SBOM work until 2025, called the AI guidance a good document but noted that many of its clusters are hard to measure, or even to define, in a consistent way across organizations.[71] An inventory standard is only as useful as the field's ability to fill it in comparably.

One further gap is worth noting. The standard has not resolved how to represent an AI system's level of autonomy.[72] That is a direct concern for edge systems, where autonomy is often the point of deployment.

Vulnerability scanning tools

Alongside the SBOM standard, practitioners already use tooling to inventory and scan the software around a model. Kenyon described using two tools together: Syft, which generates a software bill of materials by identifying the packages present in a software stack, and Grype, which scans those packages against a database of known vulnerabilities.[73] Used in tandem, they give a view of what is in the software environment and which known risks it carries. Mark, according to Kenyon, accomplishes the same task with a different tool native to AWS.[74] The specific toolchain varies by shop. The important point is that this tooling addresses the software environment surrounding the model. It does not reach the model itself. This is the same boundary the reproducibility section described, now visible in the tools practitioners actually run.

A Hardware Precedent: Blue UAS and Green UAS

The defense community has built vetted framework programs before on the hardware side, and they offer a precedent for what a comparable model framework might look like. The Blue UAS program, established by the Defense Innovation Unit in 2020, maintains a cleared list of drone platforms and components vetted against cybersecurity and supply chain requirements; Green UAS, launched by AUVSI in 2023 in partnership with DIU, extends the same standards to non-DoD users.[75] Together they give program offices a pre-cleared set of options rather than

---

[70] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026.
[71] Allan Friedman, quoted in “Global Cyber Agencies Issue New SBOMs for AI Guidance,” Infosecurity Magazine, May 15, 2026/ Friedman led CISA’s SBOM efforts until July 2025.
[72] Software Bill of Materials (SBOM) for Artificial Intelligence: Minimum Elements, G7 Cybersecurity Working Group, May 12, 2026. The document organizes an AI SBOM into seven clusters (Metadata, Models, Datasets Properties, System Level Properties, Key Performance Indicators, Security Properties, and Infrastructure); none represents a system’s level of autonomy.
[73] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026. (Syft and Grype are Anchore's open-source SBOM-generation and vulnerability-scanning tools; the interview transcript renders them phonetically.)
[74] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026. (Syft and Grype are Anchore's open-source SBOM-generation and vulnerability-scanning tools; the interview transcript renders them phonetically.)
[75] Defense Innovation Unit, Blue UAS program (established 2020); AUVSI Green UAS certification program (launched 2023 in partnership with DIU). Both vet platforms and components against NDAA

requiring each office to evaluate every vendor independently. Notably, the vetting already operationalizes the bill-of-materials idea this paper argues is missing for models: certification involves a review of hardware, software, and data pathways, including a software and hardware bill of materials.[76]

No equivalent vetted list exists for AI models today. The precedent is useful mainly as a shape. It shows that a curated, centrally maintained list of approved components is an approach the defense establishment already understands and trusts. And the programs are not one-time gates: inclusion is not permanent, and a platform can be removed if it stops meeting requirements or undergoes significant unapproved changes.[77] That continuous character is the part most relevant to models, and also where the transfer is least certain. A model that is fine-tuned, quantized, and updated in the field changes far faster and more opaquely than an airframe does, and whether a cleared-list model could keep pace with that rate of change and re-verify each altered version is an open question.

Test, Evaluation, and Red-Teaming

The last effort is less a new tool than an existing discipline that has not yet been fully extended to AI. Within the Department of Defense, the Directorate of Operational Test and Evaluation (DOT&E) has for many years subjected hardware, and increasingly software, to formal test and evaluation, including dedicated independent red-teaming to find vulnerabilities before fielding.[78] Shanahan's assessment is that this discipline has not yet been applied to AI systems at the same level of rigor or investment; in his words, the field needs to invest far more in red-teaming, and he does not see it happening yet.[79] He also stressed that test and evaluation and red-teaming need to be performed by separate entities yet must be inextricably linked, so that everything red-teaming discovers feeds directly back into model fixes and new TEVV.[80]

This matters because, as the field-disagreements section established, evaluation against a representative dataset is currently the only reliable way to detect a compromised model. If evaluation is the control that works, then underinvestment in test and evaluation is underinvestment in the one defense the field agrees on. Unlike the other efforts in this section,

---

supply chain and cybersecurity requirements; Green UAS certification includes a software and hardware bill of materials (SBOM/HBOM) review.

[76] Defense Innovation Unit, Blue UAS program (established 2020); AUVSI Green UAS certification program (launched 2023 in partnership with DIU). Both vet platforms and components against NDAA supply chain and cybersecurity requirements; Green UAS certification includes a software and hardware bill of materials (SBOM/HBOM) review.

[77] Administration of the Blue UAS Cleared List transitioned from DIU to the Defense Contract Management Agency (DCMA) in 2025; platforms can be removed for unaddressed cybersecurity issues or significant unapproved design changes.

[78] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026. DOT&E is the Directorate of Operational Test and Evaluation.

[79] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026. DOT&E is the Directorate of Operational Test and Evaluation.

[80] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026. DOT&E is the Directorate of Operational Test and Evaluation.

the institutional machinery here already exists and has a decades-long track record; what has not yet happened is its extension to AI at the scale the problem requires.

Where these leave the problem

Taken together, these efforts cover parts of the problem and leave others open. The SBOM standard and scanning tools inventory the software environment well and are beginning to reach the data layer, but they record declarations rather than confirm them, and they do not detect poisoning. The vetted-framework precedents offer a structure for certification, and a centralized model repository with a certification seal has been proposed as the AI-side equivalent, but both raise the unresolved question of who sustains that certification as models multiply. Test and evaluation is the control most likely to catch a real compromise, and it is also the one most in need of investment. No single effort closes the visibility gap.

What these approaches share is a common dependency: each works only when someone is accountable for running it, keeping it current, and acting on what it shows. An SBOM no one audits, a cleared list no one maintains, an evaluation regime no one funds: each fails in the same way, not because the mechanism is wrong but because no one owns it. That shared dependency is the subject of the next section.

9. Governance & Residual Risk

The previous section described tools and frameworks that address parts of the supply chain problem. None closes the visibility gap completely, and this section starts from an uncomfortable premise: for AI systems, complete technical visibility is not achievable today. The data cannot be fully reconstructed. The weights cannot be fully audited. The dependency chain cannot be fully traced. This is not a temporary state that better tooling will soon resolve; it is a property of how these systems are built. If governance waits for full visibility before acting, it will wait indefinitely.

At the same time, adoption cannot pause. New models are released constantly, and the organizations that build on them are under competitive and operational pressure to keep moving. Standing still is not a neutral choice. A program that refuses to adopt until the supply chain is fully transparent will fall behind one that accepts some uncertainty and proceeds. The pace that drives the visibility problem, described in the "Why Now" section, also makes waiting its own kind of risk.

Together, these two facts reframe what governance is for. If full visibility is not available and waiting is not an option, then the job of governance is not to demand complete transparency before acting. It is to decide, deliberately, which risks are acceptable to carry into deployment and which are not. The question shifts from "can we see everything?" to "do we understand enough to make an informed decision about what we are accepting?" That is a different standard, and a reachable one.

Shanahan described the decision in those terms. In his framing, supply chain risk in AI is ultimately a risk-based decision: a commander or program authority needs to understand the risk well enough to accept or reject it knowingly. He named four factors to weigh in that judgment, saying he keeps "context, complexity, urgency, and risk" in mind whenever he thinks about it.[81] The weight of each shifts with the situation. He drew the contrast sharply through the case of Ukraine, which, facing an existential threat, has no choice but to accept a level of supply chain risk that a peacetime program would not. The same model carries the same technical uncertainty in both cases. What changes is the urgency and context against which that uncertainty is weighed, and therefore the decision a reasonable authority would reach.

This is also why the timing of that judgment matters. Shanahan's position is that these questions have to be answered at the front end, before deployment, because answering them afterward means the risk has already been accepted by default. In his words, once you try to do it after fielding, "you've already bought the risk. It's already there. It's already inherent."[82] A decision not made deliberately is still a decision. It is simply one made by omission, and discovered later.

The consequences of getting that judgment wrong are not symmetric with ordinary software, and Kenyon's account of remediation shows why. If a fielded model is later found to be

[81] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.
[82] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.

compromised, there is no patch cycle. In his description, the model is removed from use, and the discovery does not lead to a fix; it leads to a national security investigation, because a poisoned model operating inside a government system is a cyberattack against the government, not a bug.[83] Kenyon was concrete about this: at a defense contractor, a compromised model would be "consumed by a three-letter" agency to remediate and investigate, and the continuous-retraining relationship would simply be shut down, without the contractor necessarily being told why.[84] The practical implication reinforces the case for front-loaded judgment. When the failure mode is investigation rather than repair, the decision made before deployment carries almost all the weight, because there is little meaningful action available after.

Even a governance process that accepts this framing faces a set of specific questions that currently have no settled answers. Shanahan laid these out directly, describing them as questions he has been thinking about and has not seen evidence the government is systematically working through. They concern who controls what, and who is accountable when a system changes:[85]

Which parts of the stack remain under government control, and which remain under vendor control? What happens when a model changes after deployment? Who validates updates, including fine-tuning pipelines, tooling, and agentic wrappers? What level of third-party assessment is needed at each layer of the stack? How should the Department of Defense treat open-weight models differently from closed or hosted models? What are the security implications of relying on commercial inference infrastructure? And what governance requirements apply, and who sets them?

Shanahan assessed that these need to be answered on the front end, and that answering them after deployment means the risk has already been taken. He was candid about the limits of his own visibility into whether the government is addressing them, noting that if his former organization is working through these questions, it is not doing so publicly. The point of listing them here is not to answer them. It shows that an adequate governance framework does not yet exist, and that the gaps are specific and nameable rather than vague.

Two structural pressures sit underneath all these questions and make them harder to answer. The first is that the complexity of modern AI systems may have outrun the capacity of manual review. Shanahan's assessment was blunt: the level of complexity has exceeded what people can assess by hand, from the chip level up through training-data poisoning and analyzing AI supply chain risk at scale may itself require AI tools.[86] A governance process built on human review alone may not scale to the number of models and the depth of each one. The second is institutional speed. Shanahan cited a concrete example: Claude running on the Maven Smart System was at least two model versions behind the current release because the newer version

[83] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026. The specific contractor named in the interview is generalized here to "a defense contractor."
[84] Author interview with Steve Kenyon, Lead AI Research Engineer, Latent AI, July 1, 2026. The specific contractor named in the interview is generalized here to "a defense contractor."
[85] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.
[86] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.

had not yet been accredited (largely due to the requirements of classified system accreditation processes).[87] The accreditation process that provides oversight is also what holds the system back, and the gap between what is available and what is approved widens as release pace increases. He framed this as an organizational design problem, a policy problem, a governance problem, and a security problem all at once.

This paper stops short of prescribing a governance model and does so deliberately: its aim is to describe the conditions clearly enough that the people accountable can design the response their own context requires. The residual risk is real: even with the best available tooling and a deliberate acceptance process, a fielded AI system carries uncertainty that cannot presently be eliminated. What a governance framework can do is make that uncertainty visible to the person accountable for the decision, so that the risk is carried knowingly rather than inherited blind. The difference between those two positions, knowing versus not knowing what has been accepted and who accepts what risks–is the difference this entire paper has been describing. It is also the one part of the problem that is available to act on right now, without waiting for a standard that does not yet exist.

[87] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.

10. Implications for Developers, Industry, Policy

The preceding sections described the visibility gap and traced where it comes from. This section turns to who inherits it. The same gap does not land the same way on everyone: it reaches the developer as a missing tool, the industry as a market pressure, and the policy apparatus as an unregulated boundary. What follows lays out how each group encounters the problem and the positions others have taken on it. It does not argue for one response over another. That judgment belongs to the people who must weigh cost, maturity, and mission fit for themselves.

For developers

The first implication is the quietest, because it is an absence rather than an event. Across the technical interviews conducted for this paper, supply chain provenance had never formally entered the development workflow. The people building these systems were not overlooking a step they knew to take; the step was never part of the process to begin with. Where supply chain risk does surface in practice, it surfaces reactively, through vulnerability management after a component is already in use, rather than through any tracking of where a model and its parts came from.

This is a matter of how the workflow was built, not of anyone's diligence. The tooling that developers rely on was assembled for a world in which the model itself was not treated as a component with a history. Two of the technical interviewees put concrete numbers to how much of their stack they cannot fully see: by their accounts, open source made up more than 80 percent of the container stack in one case and between 50 and 70 percent of a project in the other, with transitive dependencies, the dependencies of dependencies, described as the primary blind spot.[88] The gap here is the same structural opacity the paper has described throughout, arriving at the level of the individual builder: a developer can be careful, competent, and still unable to account for most of what is inside the thing being shipped. The absence of a place in the workflow to ask the provenance question is itself the exposure.

For industry

The second implication operates at the level of the market rather than the individual build, and it belongs to Shanahan, who framed it directly. Asked what internal supply chain risk is created when only a handful of vendors can affect the models running across the Department of Defense, Shanahan's concern was concentration: a market in which two or three frontier

[88] Author interviews with Mark Griffin, (Director of Software Architecture) and Abelardo Lopez (Director of Compiler and Embedded Systems), Latent AI, 2026. The open source proportions (80%+ of the container stack; 50–70% of a project) are the interviewees' own estimates of their respective stacks.

vendors hold that much influence amounts, in his assessment, to a monopoly position, and one he did not consider healthy for the government that depends on it.[89]

Shanahan connected this to a second dynamic affecting the smaller specialized builders. He had seen companies emerge offering to tune a narrow model for a specific government use, and considered the idea a sound one, but expected many to struggle as frontier models improve. His reasoning was that the frontier models are becoming capable enough to be applied to new data without much tuning, which erodes the space a fine-tuning-focused startup was built to occupy. Based on feedback from people across the tech industry, he now expects the government will turn to hybrid architectures, in which organizations will route different tasks to different open and closed models. Government organizations may turn to frontier models when they are genuinely better (usually for the most complex, compute-intensive tasks), open models when they are good enough, and fine-tuned internal models or RAG when control, cost, or classified data matter most. These are his assessments of where the market is heading, offered as a forecast rather than a settled fact, and recorded here as such.

For policy

The third implication is the one that sits closest to the surface of current debate, and it is where the visibility gap becomes a question of rules rather than tooling. The clearest articulation of it comes from Morgan Plummer, who served as managing director of the Defense Department's National Security Innovation Network and as a professor of practice at the U.S. Air Force Academy, writing in DefenseScoop.[90] Plummer's argument is that rather than statutory guardrails set by Congress, U.S. policy relies largely on general Pentagon guidance calling for "appropriate levels of human judgment" in military systems, language that leaves the operative questions unanswered.[91]

The gap Plummer describes has a structural cause. The systems in question are increasingly built not inside traditional defense contractors but in startups, research labs, and technology firms whose primary markets are commercial, and which have little history working with the defense establishment.[92] When their products are adapted for national security purposes and no clear rules govern that adaptation, the boundaries end up negotiated in real time. Plummer's proposed response is to use the acquisition system as the lever: federal acquisition regulations

[89] Author interview with Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.), July 10, 2026.
[90] Morgan Plummer, “A Policy Gap Is Threatening the Pentagon’s AI Innovation Pipeline,” DefenseScoop, March 26, 2026. Plummer is Vice President of Policy at Americans for Responsible Innovation; he previously served as Managing Director of the DoD National Security Innovation Network and as a Professor of Practice at the U.S. Air Force Academy.
[91] Morgan Plummer, “A Policy Gap Is Threatening the Pentagon’s AI Innovation Pipeline,” DefenseScoop, March 26, 2026. Plummer is Vice President of Policy at Americans for Responsible Innovation; he previously served as Managing Director of the DoD National Security Innovation Network and as a Professor of Practice at the U.S. Air Force Academy.
[92] Morgan Plummer, “A Policy Gap Is Threatening the Pentagon’s AI Innovation Pipeline,” DefenseScoop, March 26, 2026. Plummer is Vice President of Policy at Americans for Responsible Innovation; he previously served as Managing Director of the DoD National Security Innovation Network and as a Professor of Practice at the U.S. Air Force Academy.

could require that, in order to contract with the department, AI labs meet baseline safety and governance standards and document their intended use cases before operational integration.[93] This is a call to action, and it is his. The paper notes that the lever exists and that a defense insider is proposing it; it does not itself recommend that the lever be pulled.

One gap, three vantage points

Set side by side, the three implications are not separate problems but one condition seen from three positions. The developer meets it as a question the workflow was never built to ask. The industry meets it as pressure toward concentration, with the specialized builders most exposed. The policy apparatus meets it as a boundary that no standard yet governs, left to be settled case by case. In each case the risk is inherited rather than chosen, and inherited quietly, which is the through-line the paper has followed from the start. The map is legible in hindsight and hard to read at the moment. What the following section returns to is why the moment to read it is the one available now.

[93] Morgan Plummer, “A Policy Gap Is Threatening the Pentagon’s AI Innovation Pipeline,” DefenseScoop, March 26, 2026. Plummer is Vice President of Policy at Americans for Responsible Innovation; he previously served as Managing Director of the DoD National Security Innovation Network and as a Professor of Practice at the U.S. Air Force Academy.

11. Recommendations

Earlier sections established that a complete provenance record is not one document but several: a binary bill of materials that existing tooling handles reasonably well, a data bill of materials covering what a model was trained on, and a topology bill of materials describing how a deployed system is wired together. The data and topology records are the ones current standards leave largely uncovered. The framework described here addresses the model and its data lineage. It is one component of that larger record, not a substitute for it.

The framework described here maps onto the data and models portions of that record, the parts existing software-supply-chain tooling reaches least well. It does not close the topology bill of materials, which Aronchick identified as a separate and largely unaddressed piece: an account of how a deployed system is wired together. At the edge that account is not incidental. As Aronchick noted, the way a model is deployed is itself part of what must be reproducible, because a system that reads from a camera at a particular angle, with particular preprocessing, depends on that pipeline as much as on the model weights. A complete AIBOM would need to record the topology alongside the data and model lineage, so that the artifact a program office reviews and the system that runs in the field can be tied to the same specification.

The AIBOM Recipe: Establishing AI Model Provenance

This section sets out what an Artificial Intelligence Bill of Materials (AIBOM) framework would need to include to establish a model's provenance. It draws on work Latent AI developed under an Army xTech Scalable AI2 and Army Phase 2 SBIR.

An AIBOM is a structured record of the configurations involved in building and validating an AI model, intended to make a model and its data traceable across the parties that build, adapt, and deploy it. For defense procurement, that traceability is the condition the preceding sections identified as absent. The more completely an AIBOM captures those configurations, the more a model's lineage becomes visible to the party accountable for fielding it.

Core Components of an AIBOM

Within the data and model portion of that larger record, an effective AIBOM would need at least three elements to track the software, hardware, and data that go into a model: a software bill of materials (SBOM) detailing the supply-chain relationships and specific software components used to build, test, and validate the AI model; a record of model details capturing the model's architecture, inherent properties, training data parameters, hyperparameters, and intended use cases; and a data lineage tracing the historical pedigree and origin of all data utilized in the model's creation.

The Need for Standardized Terminology

Terminology across the current MLOps ecosystem is fragmented, with terms such as "model cards," "manifests," "recipes," and "Data Version Control" used to mean different things. Consistent terminology is a precondition for tracking a model accurately across parties. Naming

a model only as “YOLOv5,” for example, does not identify it precisely enough to reproduce. To be precise, an AIBOM would need to account for:

- Source Variants: Different iterations from the open-source community.
- Model Configurations: Specific sizes such as nano, small, medium, or large.
- Composite Architectures: The use of pretrained backbones (e.g., a RESNET backbone paired with a distinct detector head).

Achieving Reproducibility Through "Recipes"

Reproducibility is the standard an AIBOM would need to reach: the ability to rebuild a model and confirm that what runs in the field came from a known set of source materials. Reaching it takes more than an inventory of parts. Latent AI describes this as a recipe: the pre-configured assets together with the step-by-step instructions required to build, deploy, and maintain an optimized model. An inventory names the ingredients; a recipe adds the method by which they are combined, which is what makes the result reproducible rather than merely documented.

Each recipe corresponds to a specific task, such as object detection or classification, and is tied to a designated model and hardware target. Because it records the optimizations applied for a particular device, it attributes a deployed artifact to the specific steps that produced it rather than to the model in the abstract.

Lifecycle Management and Security

A model does not stay fixed once fielded; it is restrained, fine-tuned, and adapted over time. An AIBOM that recorded only the model as first built would fall out of step with what is deployed. To stay useful, it would need to track derivatives and variants as a model evolves and ingests new data.

A record of this kind also has a security use. If a model is later found to be compromised through data poisoning or adversarial manipulation, a documented AIBOM gives engineers the reference points needed to trace the affected version and return the system to a known, prior state.

Forensics and Model Registries

The same record supports forensic analysis. When model components are searchable, an organization can identify which fielded models share a compromised dataset or architecture, rather than assessing each model in isolation.

For that to be possible at scale, latent AI points to two pieces of supporting infrastructure: a national model registry, a centralized and searchable index of AIBOM contents through which a fielded model can be located by its components; and automated ingestion tooling to populate AIBOMs for models already in use, so that models fielded before the framework existed can be brought into it rather than left unrecorded.

12. Conclusion

This paper has followed one gap from four positions. It began with where the gap comes from, in the pace of adoption outrunning the documentation of what is adopted. It then took the edge AI system apart to show what the gap looks like in the assembly, and where the edge-specific tail separates the model that is reviewed from the model that runs on the device. From there it sets out the seven risk areas the gap opens, which sort into what cannot be seen about a model in motion and what no tooling reveals because it is not a technical property of the artifact. And it described who carries the gap: the developer, for whom provenance is a step the workflow was never built to include; the industry, where the pressure runs toward concentration; and the policy apparatus, a boundary no official standard yet governs.

The condition underneath these sections is the one stated at the outset. The gap is structural rather than a failure of effort: a model's training data cannot be read back out of the finished model, and its weights are not human-readable. The joint guidance issued by the NSA, CISA, and the FBI in May 2025 uses the word opacity to describe web-scale training datasets, noting that their scale and opacity make them especially difficult to audit or govern. Opacity is visibility’s direct opposite, and visibility is the precondition for every security control and mitigation plan built on top of it. You cannot secure, verify, or make a sound risk decision about what you cannot see.

The mechanism that pushes visibility out of view is competitive rather than adversarial. Organizations adopting AI are moving quickly, and the speed is itself a form of risk avoidance: the perceived cost of falling behind outweighs, in the moment, the diffuse and unquantified cost of not knowing what went into a given model. The same posture that looks like prudence is what pushes questions of provenance, licensing, and modification history out of view. The dependency accumulates one component at a time, none of it flagged at the moment of purchase, clear only in retrospect.

What distinguishes the present moment is that the reckoning has not been forced. As Section 2 set out, entities like Ukraine have no choice but to accept a level of supply chain risk that a peacetime program would not, because they are operating under existential urgency and moving so fast that steps are left off and reconstructed after the fact. The United States retains the option to set the bar for evaluation before that pressure arrives. The conditions that make the supply chain hard to see are also the conditions under which it can be examined without a crisis dictating the terms. This is a statement about which moment offers the clearest view, not a claim that time is running out.

Complete technical visibility is not available at that moment or at any moment presently. As Section 9 set out, the data cannot be fully reconstructed, the weights cannot be fully audited, and the dependency chain cannot be fully traced, and this is a property of how these systems are built rather than a temporary state that better tooling will resolve. A governance process that waits for full visibility before acting will wait indefinitely, and a program that refuses to adopt until the supply chain is fully transparent will fall behind one that accepts some uncertainty and proceeds. In Shanahan's framing, supply chain risk in AI is a risk-based decision, and a commander or program authority can make that decision only if the risk is understood well

enough to accept or reject it knowingly. He placed that judgment at the front end, because attempting it after fielding means, in his words, the risk has already been bought.

What a governance framework can do now is make the residual risk visible to the person accountable for the decision, so that it is carried knowingly rather than inherited blind. That difference is the one part of the problem available to act on in the present, without waiting for a standard that does not yet exist. The view is hardest to hold at the edge, where the model is transformed after it is inspected, placed where it cannot easily be reached, and cut off from the update path that would allow correction, and it is available to hold now.

Appendix

Acknowledgements

Lt. Gen. John "Jack" N.T. Shanahan, USAF (Ret.) — inaugural Director, Joint Artificial Intelligence Center (JAIC); founding director of Project Maven. Interviewed July 10, 2026.

Jan Ernst — Director of AI, Latent AI. Interviewed June 23, 2026.

Steve Kenyon — Lead AI Research Engineer, Latent AI. Interviewed July 1, 2026.

Abelardo Lopez — Director of Compiler and Embedded Systems, Latent AI. Interviewed June 17, 2026.

Chris Ritter — Chief Federal Application Architect, Dell Technologies (Federal). Interviewed July 14, 2026.

David Aronchick — CEO, Embedl. Interviewed July 15, 2026.

Mark Griffin — Director of Software Architecture, Latent AI. Interviewed June 18, 2026.

Sources

Government and multilateral guidance

- National Security Agency, Cybersecurity and Infrastructure Security Agency, and Federal Bureau of Investigation, et al. *AI Data Security: Best Practices for Securing Data Used to Train & Operate AI Systems.* Joint cybersecurity information sheet, May 22, 2025.
- G7 Cybersecurity Working Group. *Software Bill of Materials (SBOM) for Artificial Intelligence: Minimum Elements.* May 12, 2026. Building on the G7 "Shared Vision on SBOM for AI" (June 2025). Published with CISA and G7 partners.
- U.S. Department of Defense. Directive 3000.09, *Autonomy in Weapon Systems.* Updated January 2023.
- Defense Innovation Unit. Blue UAS program (established 2020). Administration of the Blue UAS Cleared List transitioned from DIU to the Defense Contract Management Agency in 2025.
- Association for Uncrewed Vehicle Systems International (AUVSI). Green UAS certification program (launched 2023, in partnership with DIU).

Reports and academic sources

- Nikhil Mulani. *An Evolving AI Supply Chain.* University of California, Berkeley, October 2025.
- Leonardo Gambacorta and Vatsala Shreeti. *The AI Supply Chain.* BIS Papers No. 154. Bank for International Settlements, March 2025.

- Nicholas Carlini et al. "Poisoning Web-Scale Training Datasets Is Practical." 2024 IEEE Symposium on Security and Privacy. arXiv:2302.10149.
- Pablo Villalobos et al. "Will We Run Out of Data? Limits of LLM Scaling Based on Human-Generated Data." arXiv:2211.04325.
- S. Alemohammad et al. "Self-Consuming Generative Models Go MAD." arXiv:2307.01850.
- Shiyin Ouyang et al. "Position: Sora Shows That Data Is More Important Than Compute." arXiv:2503.14023.
- Gustavo Correa Publio et al. "ML-Schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies." arXiv:1807.05351.

News and commentary

- Paul Mozur and Valerie Hopkins. "Ukraine's War of Drones Runs Into an Obstacle: China." *New York Times*, September 30, 2023.
- "China's Drone War in Ukraine." *The Diplomat*, January 26, 2026. The 97 percent figure is the outlet's reported estimate, attributed to industry sources.
- Morgan Plummer. "A Policy Gap Is Threatening the Pentagon’s AI Innovation Pipeline." *DefenseScoop*, March 26, 2026.

Model cards and repositories

- Florence-2-large model card. Hugging Face, microsoft/Florence-2-large (MIT license).
- OWLv2 model cards. Hugging Face, google/owlv2-base-patch16 and google/owlv2-large-patch14-finetuned (Apache 2.0 license).
- "weapon-detection-app." Hugging Face Space, user marcuscanhaco.

Corporate and legal filings

- SolarWinds Corp. Form 10-K (FY2020) and SEC disclosures. MITRE ATT&CK Campaign C0024 (SolarWinds Compromise).
- Anthropic–DoD prototype agreement (July 2025), supply-chain-risk designation (letters dated March 3, 2026), and Anthropic's federal lawsuits challenging the designation (filed March 9, 2026). The litigation was ongoing as of this paper's research and its outcome is not reflected here.

Glossary

SLSA (Supply-chain Levels for Software Artifacts)
A framework for build integrity and provenance. Adds verifiable guarantees about how a software artifact was built. Like SBOM, it was built for the binary and the pieces surrounding the model, not for the model itself.

Binary BOM
An inventory of the software artifact and its surrounding pieces. Reasonably handled by existing tooling.

Data BOM
An account of what data went into the model. Aronchick described the data as an "utter black box," not introspectable, where provenance and lineage sit largely untouched by current standards.

Topology BOM
An account of how the pieces of an AI system are wired together and deployed. Described by Aronchick as poorly handled by current tooling.

AI BOM

A structured record of the configurations involved in building and validating an Ai model, intended to make a model and its data traceable across the parties that build, adapt, and deploy it.

Chainguard
One of several companies that build tooling and products to help organizations meet SLSA levels.

Fine-Tuning

Taking a foundation model and specializing it for a specific domain by feeding in new data. This is where a customer's or integrator's own data enters the pipeline, and where provenance questions compound.

Optimization and Compilation
Taking the original model as it comes from the ML framework (PyTorch, TensorFlow, and similar) and transforming it into an intermediate representation that can be manipulated to produce an artifact with lower latency, a smaller memory footprint (for example through

quantization), or both. The optimization usually targets a specific hardware device and takes advantage of any capabilities present on that target, such as hardware accelerators. Put more plainly, it is making the model run better, faster, more accurately, or at lower power, on resource-constrained devices. The optimization tooling itself has its own dependencies, and the transformation is rarely tracked.

Reproducibility

A deterministic way to rebuild what was built: to be able to say unequivocally that the thing that ran came from a specific, verified set of source materials.

Frontier Model

The largest and most capable general-purpose AI models at the leading edge of current capability, typically trained at very large scale by a small number of well-resourced developers and usually delivered as closed or hosted services. In this paper the term marks a contrast with the smaller, task-specific computer-vision and language models deployed at the edge: frontier models are the upstream systems from which many edge models are derived, not the models that run on the device. The distinction matters because the visibility a program office once held over a task-specific vendor model (as in the pre-frontier Project Maven period) has narrowed sharply for models derived from frontier systems, whose training data and internals are not disclosed.