FORGE

By LAVA

The Top 10 Data Center And AI Infrastructure Security Risks

Harden The Metal Beneath The Model.

A targeted, actionable framework for the teams building, running, and buying, AI compute, neocloud providers, security architects, and CISOs.



Acknowledgment

Built to evolve with the field, and kept open so the whole AI and security community can use it, challenge it, and improve it.

With thanks to the practitioners and reviewers below,
whose feedback shaped this framework as it took form

Have feedback or want to contribute?

research@lavalabs.io

Authors

Michael Katchinskiy
Head of Security Research, Lava

Yakir Kadkoda
CTO, Lava

Reviewers

Amir Jerbi, Former CTO, Aqua Security

Arthur Reed, Security Engineer, Pacific Northwest National Laboratory

Assaf Morag, Individual Contributor

Assaf Namer, Head of AI security, Google Cloud

Bill Stout, Former Technical Director, AI Product Security, ServiceNow

Dan Farmer, Individual Contributor

Daniel Iziourov, Platform Security, Nebius

Erik Deumens, Director Research Computing, University of Florida

Florina Ciorba, Associate Professor, Head of High Performance Computing Group, University of Basel under Erik Deumens

Golan Ben-Oni, CIO/CISO, IDT

Guy Bilitski, Leading AI Operations, SDS AI

Guy Shanny, Co-Founder & CEO, Polar Security, acquired by IBM

Michael Bargury, CTO, Zenity

Roey Yaacovi, CTO, DSPM & AI Security, IBM

Saad Malik, CTO, Spectro Cloud

Selim Aissi, Former Vice President, Global Information Security, Visa & Intel

Tony Rea, Global AI Infrastructure Lead, Dell Technologies

Tyson Macaulay, Deputy Director ,National Centre For Critical Infrastructure Protection, Security and Resilience

Vjaceslavs Klimovs, Senior Technical Director, Roblox

Ziv Karliner, CTO, Pillar Security

Table Of Contents

Acknowledgments

Executive Summary 03

1.1 What This Framework Covers

1.2 Who This Framework Is For

1.3 The Forge Top 10 At A Glance

Chapter 1: Introduction 04

2.1 Why AI Data Centers Infrastructure Security Is Different

2.2 Threat Model

2.3 How The Risks Were Selected

Chapter 2: The Forge Framework

Forge-01 Firmware & Hardware Integrity Compromise 9

Forge-02 Network & Interconnect Vulnerabilities 12

Forge-03 Unsafe Multi-Tenant Isolation And Resource Reuse 14

Forge-04 Insecure Out-Of-Band Management Plane 17

Forge-05 AI Infrastructure Supply Chain Compromise 20

Forge-06 Insecure Facility And Datacenter Management Systems 23

Forge-07 Insecure Data And Artifact Handling 25

Forge-08 Certification Gaps And Provider Transparency Failures 27

Forge-09 Insecure Operational Infrastructure Services 29

Forge-10 Vendor Embargo Gaps And Patch Velocity Failures 31

Versioning And Contributions

WHAT IS FORGE

A security framework for AI Data Centers

A practical framework for identifying and reducing the most important risks in the infrastructure layer that trains, hosts, and serves AI.

WHY FORGE?

AI security needs to go below the model.

FORGE focuses on the infrastructure layer where hardware, fabrics, management planes, and provider controls determine trust.

HOW IT IS USED

Providers harden. Customers evaluate.

FORGE gives both sides a shared language for assessing GPU clouds, neoclouds, AI datacenters, and internal AI clusters.

Executive Summary

AI datacenters are being built faster than they are being secured.

The rapid expansion of datacenters, GPU clouds, and specialized compute environments has introduced security risks across hardware, networking, storage, orchestration, identity, management planes, and physical operations. Many of these risks resemble traditional datacenter or cloud security issues, but modern data centers and AI infrastructure changes their severity: systems originally designed for trusted operators are now supporting high-value, multi-tenant workloads from unrelated customers.

The **Top 10 Data Centers & AI Infrastructure Security Risks** provides a practical framework for identifying, prioritizing, and reducing the most important security risks in the infrastructure layer that powers AI. The framework defines the most critical failure modes in AI infrastructure and helps translate them into concrete security requirements.

What This Framework Covers

This framework focuses on the security of AI infrastructure and the datacenters that house it: the physical hardware, networking fabrics, management planes, orchestration systems, storage systems, and operational environments on which AI workloads run.

It does not focus on AI models themselves or application-layer risks such as prompt injection, insecure agent behavior, model abuse, or model-level evaluation. Those risks are addressed by frameworks focused on other parts of the AI stack, including the OWASP Top 10 for LLM Applications, MITRE ATLAS, the NIST AI Risk Management Framework, and ISO/IEC 42001. Together, these resources give practitioners a more complete picture of AI security, from governance and application-layer risks down to the underlying compute infrastructure.

Some risks in this document also exist in traditional datacenter and cloud environments. They are included here because AI infrastructure makes them materially more severe: shared high-value compute, complex accelerator clusters, dense management layers, and multi-tenant operations can turn ordinary infrastructure weaknesses into more severe security risks because the likelihood and impact of any incident are much higher than in traditional enterprise level software and service deployments.

Who This Framework Is For

Neo-cloud providers can benefit from the framework as a practical guide to the AI infrastructure threat landscape, attack surface review, environment hardening, security prioritization, and maturity demonstration.

AI infrastructure customers can benefit from the framework as a practical guide for procurement, security reviews, contractual requirements, provider comparison, and assessing resilience against realistic tenant-to-infrastructure compromise.

Introduction

The rate at which the modern datacenter and AI infrastructure is being built is heavily outpacing the ability to secure it. When this infrastructure, GPU clusters, training pipelines, high-performance networking, and inference endpoints, is compromised, the blast radius is unlike conventional compute. Attackers gain access to proprietary models representing hundreds of millions of dollars, the power to poison foundational training data, and persistent footholds in the most highly privileged environments available.

This dynamic is reinforced by a structural market imbalance: GPU scarcity gives providers outsized leverage. When demand for accelerated compute far outstrips supply, customers often cannot choose their provider based on security posture - they take what is available. Providers face little market pressure to invest in security maturity, and customers accept risk they would not tolerate in conventional cloud. This imbalance is the backdrop against which every risk in this document should be read. This market pressure is becoming more dangerous as AI-enabled security research and exploitation capabilities compress the timeline for defenders. Weaknesses that might once have remained obscure for years may now be discovered, chained, and operationalized much faster.

At the same time, AI-enabled security research and exploitation capabilities are compressing the timeline for defenders: weaknesses that might once have remained obscure for years may now be discovered, chained, and operationalized much faster.

Why AI Infrastructure Security Is Different

AI infrastructure sits at the intersection of cloud computing, high-performance computing, and physical datacenter operations. It uses familiar components such as servers, storage, networks, schedulers, management planes, and identity systems, but combines them in ways that change the security model.

Traditional HPC environments were often designed for trusted users, research communities, or internal operators. Modern AI infrastructure increasingly supports commercial, high-value, multi-tenant workloads from unrelated customers. As a result, assumptions that were acceptable in trusted or single-organization environments can become serious security risks when applied to shared AI infrastructure.

Recent Reporting

Attackers have already begun targeting the infrastructure, supply chains, and datacenter ecosystems that support advanced AI and HPC workloads. In some cases, the objective is direct theft of sensitive research, models, or engineering data, in others, it is espionage, prepositioning, or the ability to disrupt strategically important compute environments. Recent reporting illustrates both patterns:

APRIL 2025

Reported attacks on U.S. AI data centers

In April 2025, TIME reported that researchers speaking with national security officials and datacenter operators learned of one case in which a top U.S. technology company's AI datacenter was attacked and intellectual property was stolen. They also described another case in which a similar facility was targeted through a specific unnamed component, had the attack succeeded, it could have taken the entire datacenter offline for months.

JANUARY 2026

Nation-state targeting of AI infrastructure IP

Federal prosecutions have confirmed that nation-states are directly targeting AI datacenter architecture at the hardware and systems level. In one case, a former engineer at a major U.S. technology company was convicted of economic espionage after stealing thousands of pages of AI datacenter designs, including chip architecture, systems integration specifications, and orchestration software for large-scale model training.

APRIL 2025

Reported exfiltration from China's National Supercomputing Center in Tianjin

In April 2026, public reporting described claims that more than 10 petabytes of data had been stolen from the Tianjin supercomputing hub, a facility that reportedly supports thousands of research, industrial, and government users. Data claimed to originate from the breach was advertised for sale on Telegram channels. The incident illustrates the concentration risk created when high-value workloads, data, and research outputs are pooled in a single advanced compute environment.

These are early examples. The attack surface is likely to expand as the stack matures, and this document will evolve alongside it.

Threat Model

The risk is best understood by looking at potential attackers. The threat model spans far more than the classic "outside attacker", and the controls in this document are calibrated against the full set.

External attacker

An unauthenticated or unauthorized actor with network access.

Compromised cloud customer

A legitimate customer account, tenant, or workload that has been compromised and is operating on the provider's shared infrastructure.

Malicious enterprise insider

An employee or contractor of the AI-consuming organization with legitimate access.

Malicious or compromised insider

An employee, contractor, technician, system integrator, logistics provider, or break-fix vendor with legitimate access to the customer or provider environment.

In-transit adversary

An actor with access to hardware or components during manufacturing, staging, warehousing, shipping, or delivery before they reach the provider's secured environment.

System integrator

An organization responsible for assembling, configuring, or shipping complete systems from individual components.

Supplier

An upstream hardware, firmware or driver vendor.

AI agents

Any AI agents acting on behalf of and impersonating any of the above.

Enterprise Compromise as an Initial Access Vector

The risks in this framework focus on AI infrastructure-specific or AI infrastructure-amplified failure modes. They do not replace the need for strong enterprise security controls. In practice, many attacks against AI infrastructure may begin through familiar enterprise compromise paths, including weak or missing MFA, phishing, credential theft, SaaS compromise, and weak identity controls.

These paths should be treated as cross-cutting initial access vectors. A compromised identity, endpoint, SaaS account, CI/CD system, or administrative credential can provide the foothold needed to reach several of the risks described in this framework. These specific sections therefore focus on what can happen once an attacker reaches, or can influence, the AI infrastructure environment itself.

How Risks Were Selected

Candidate risks were drawn from:

- Analysis of security research, incident reports, and published CVEs affecting AI-specific components.
- Review of vendor security documentation from NVIDIA, AMD, and major cloud providers.
- Observed vulnerability patterns across GPU cloud environments and ML orchestration platforms.
- Discussions with security practitioners working in AI infrastructure across multiple verticals.

Each candidate was evaluated against four dimensions, **Likelihood, Impact, Exploitability,** and **Detection Difficulty**, with the highest-scoring risks selected for inclusion.

Risks that also exist in conventional cloud and datacenter environments were included only when AI infrastructure materially increases their severity.

Versioning and Contributions

This is a living document. AI infrastructure is evolving rapidly, and so are the attacks against it. Future revisions will add new risks, refine existing ones, and incorporate lessons learned from the field. Feedback and proposed additions from practitioners are welcome and will be credited upon publication of the new version.

The Five Domains of the FORGE Lens

FORGE domains define the evaluation lens: the infrastructure areas where AI security risk lives.
The Risk Matrix below maps individual risks into these domains.

F

Fleet integrity

Trust in the hardware, firmware, software artifacts, images, dependencies, and supply paths that make up the AI infrastructure fleet.

O

Operations & management planes

Privileged systems used to control, automate, and administer AI infrastructure, including BMCs, schedulers, orchestration, automation, and admin tooling.

R

Resource isolation

The boundaries that separate tenants, workloads, execution environments, and reused infrastructure across shared AI systems.

G

Grid

The networking fabrics and facility systems that connect AI clusters and keep them powered, cooled, and operational.

E

Evidence & exposure management

The evidence customers need to understand provider security maturity, architectural scope, exposed services, and patch velocity.

The FORGE Top 10 at a Glance

FORGE-01 Firmware & Hardware Integrity Compromise

Unauthorized firmware or hardware changes can create persistent compromise below the operating system, surviving rebuilds, disk wipes, and tenant reassignment.

FORGE-02 Network & Interconnect Vulnerabilities

Weak fabric isolation, management protection, and monitoring can expose InfiniBand, RoCE, and RDMA paths to discovery, abuse, or lateral movement.

FORGE-03 Unsafe Multi-Tenant Isolation and Resource Reuse

Failures in shared or reassigned infrastructure can let tenants cross boundaries, observe residual state, discover other workloads, or degrade neighboring jobs.

FORGE-04 Insecure Out-of-Band Management Plane

Weaknesses in BMCs, management protocols, networks, and fleet tools can give attackers hardware-level control outside the host OS.

FORGE-05 AI Infrastructure Supply Chain Compromise

Unverified hardware, firmware, images, drivers, packages, models, or infrastructure artifacts can introduce compromise through trusted supply paths.

FORGE-06 Insecure Facility & Datacenter Management Systems

Weaknesses in cooling, power, DCIM, access control, or facility networks can disrupt AI workloads, expose asset maps, or bypass server-side defenses.

FORGE-07 Insecure Data and Artifact Handling

Weak controls over datasets, prompts, logs, embeddings, checkpoints, model weights, caches, backups, and deployment artifacts can enable exposure or tampering.

FORGE-08 Certification Gaps & Provider Transparency Failures

Customers may rely on certifications, attestations, or provider claims that do not clearly cover the infrastructure actually running their workloads

FORGE-09 Insecure Operational Infrastructure Services

Compromise of schedulers, control planes, provisioning workflows, automation, or admin interfaces can give attackers broad control over AI workloads and nodes.

FORGE-10 Vendor Embargo Gaps & Patch Velocity Failures

Delayed disclosure, slow patching, disruptive updates, and version drift can leave AI infrastracture exposed to known vulnerabilities.

Risk Matrix

FORGE IDs are ordered by severity, from highest to lowest. the matrix groups each risk by domain and shows its likelihood, impact, and detection difficulty.

Risk ID	Domain	Risk Name	Risk Level	Likelihood	Impact	Detection Difficulty
FORGE-01	F	Hardware & Firmware Integrity Compromise	⚠ Critical	Medium	⚠ Severe	High
FORGE-02	G	Network & Interconnect Vulnerabilities	⚠ Critical	Medium	⚠ Severe	High
FORGE-03	R	Unsafe Multi-Tenant Isolation and Resource Reuse	⚠ Critical	Low	⚠ Severe	⚠ Very high
FORGE-04	O	Insecure Out-of-Band Management Plane	⚠ Critical	Medium	High	⚠ Very high
FORGE-05	F	AI Infrastructure Supply Chain Compromise	⚠ Critical	High	High	High
FORGE-06	G	Insecure Facility & Datacenter Management Systems	High	Low	High	⚠ Very high
FORGE-07	R	Insecure Model Artifact Handling	High	High	High	Medium
FORGE-08	E	Certification Gaps & Provider Transparency Failures	High	Medium	High	Medium
FORGE-09	O	Insecure Operational Infrastructure Services	High	High	Medium	Medium
FORGE-10	E	Vendor Embargo Gaps & Patch Velocity Failures	Medium	Medium	Medium	Low

F Fleet integrity O Operations & Management Planes R Resource isolation G Grid E Evidence & Exposure Management

Forge-01 Firmware & Hardware Integrity Compromise

Definition

Firmware & Hardware Integrity Compromise refers to unauthorized or attacker-controlled modification of firmware across system components such as UEFI, BMC, NICs, GPUs, and storage controllers. Because this code runs before or below the OS, compromise can persist across OS reinstalls and disk wipes, evade many OS-level security tools, and survive tenant reassignment in shared bare-metal infrastructure.

Description

Firmware and hardware integrity matters because these components sit beneath the operating system and outside the visibility of most normal security tools. A compromised firmware component can affect how a system boots, how devices behave, and what the OS itself is allowed to see.

This makes trust difficult to establish after the fact. Providers need a way to prove that a platform booted into an expected, untampered state, and that critical components have not drifted from approved versions. Controls such as Secure Boot, Measured Boot, signed firmware enforcement, and cryptographic attestation help provide that assurance, but only when they are present, enabled, correctly configured, monitored, and independently verified.

The risk is especially important in AI infrastructure because these systems are valuable, shared, and often reassigned between tenants. In bare-metal environments, a malicious tenant with sufficient local access may attempt to abuse firmware update paths or device control interfaces before returning the node. If signed-firmware enforcement, attestation, and reassignment checks are absent, disabled, or unverified, the next tenant may inherit a compromised machine.

The BMC is another important path because it can control power, remote console access, virtual media, and firmware update workflows independently of the host OS. If BMC access is exposed through weak credentials, poor segmentation, or vulnerable management interfaces, an attacker may be able to modify firmware or establish persistence below the operating system.

This threat is not theoretical. **BlackLotus** showed that UEFI bootkits can execute before the OS loads, and **CosmicStrand** showed that motherboard firmware implants can persist across OS reinstalls and raise concerns around resale or refurbishment channels.

See also: FORGE-04 Insecure Out-of-Band Management Plane for management plane attacks that can lead to firmware compromise.

Impact and Failure Modes

Firmware cannot be verified

Attestation and Measured Boot are the features that let a provider prove a machine's firmware hasn't been modified. Where they aren't enabled, providers have no cryptographic proof that the BIOS, BMC, NIC, GPU, or storage firmware can be trusted. The firmware may well be fine, but there is no way to tell. On hardware that supports these features, the gap is usually operational, the feature is off, unconfigured, or unmonitored, rather than a hardware limitation.

Firmware update paths are exposed or weakly validated

Firmware is updated through vendor flashing tools, BMC update functions, and maintenance workflows. If these paths are reachable without proper authorization, or do not enforce signature checks, an attacker may install firmware that survives normal rebuilds and is difficult to remove.

Cross-tenant persistence in bare metal

In bare-metal reuse, a malicious tenant with sufficient local or device-level access may abuse firmware or privileged control interfaces before returning a node. If the provider only wipes the OS and does not reflash, or verify firmware state, the next tenant may inherit a compromised machine.

Attestation signals can be missing or misleading

BMC platforms frequently ship with integrity features turned off. HPE iLO's Global Component Integrity, for example, is disabled by default and requires a full power cycle to enable. Once on, the green checkmark next to a device only confirms the device is genuine, not that its firmware is untampered. Operators who treat the checkmark as proof of trusted firmware develop a false sense of assurance.

Attestation coverage is uneven across mixed fleets

Hardware-backed attestation depends on platform and component support, such as an external Root of Trust, eRoT, that must be designed into the system. NVIDIA datacenter products have broad eRoT coverage, but support across CPUs, NICs, storage controllers, BMCs, and older platforms may vary significantly. Providers cannot assume every component in a mixed fleet can produce useful attestation evidence.

Attestation evidence needs reference values to be meaningful

An attestation measurement only proves what value a device reported. To determine whether that value is trusted, the verifier needs vendor-published reference values for the specific device and firmware version. CoRIM, Concise Reference Integrity Manifest, is the emerging IETF standard for publishing this evidence, but adoption remains limited. NVIDIA is currently the main hardware vendor publishing CoRIM data for its devices.

Prevention and Mitigation Strategies

For providers and datacenter operators

1. **Establish a hardware root of trust:** Use a hardware or silicon root of trust, together with a TPM or equivalent, to anchor platform identity, record boot measurements, and support attestation. Where supported, use silicon root-of-trust efforts such as Caliptra for chip identity, measured boot, and attestation.

2. **Enforce trusted boot and signed firmware:** Enable Secure Boot and Measured Boot, require signed firmware, and prevent unauthorized firmware or boot components from running.

3. **Verify firmware before production use, and plan for partial coverage:** Where supported, use device attestation protocols such as SPDM to validate firmware measurements against vendor-published reference values. Confirm that attestation features are enabled and that reported status reflects firmware integrity, not only device authenticity. For components without attestation coverage, use compensating controls such as re-flashing to known-good images when systems are received or reassigned.

4. **Monitor firmware state across the fleet:** Maintain a continuously updated inventory of firmware versions for components such as BIOS, BMC, GPU, NIC, and storage devices. Treat unexpected firmware changes, baseline drift, or failed or changed attestation results as security events requiring investigation. Alert when nodes deviate from approved firmware baselines, when attestation results change between boots without a corresponding maintenance event, or when attestation fails entirely. Without fleet-level visibility, a single tampered node can remain in production indefinitely.

5. **Gate sensitive workload placement on attestation:** Integrate remote attestation with schedulers so sensitive workloads run only on verified nodes, and automatically quarantine systems that fail verification.

6. **Secure firmware lifecycle operations:** Restrict access to firmware tooling and management interfaces, require approved update workflows, and re-attest systems during intake, servicing, and tenant transitions.

7. **Maintain secure recovery procedures:** Keep known-good firmware images, documented recovery workflows, and replacement criteria so compromised platforms can be restored or removed from service with trust re-established.

For customers evaluating a provider

How much you can check yourself depends on the offering. On bare-metal or dedicated nodes you can verify the first two directly.

1. **Check that trusted boot is actually on (bare metal offering):** From the OS, confirm Secure Boot is enabled and a TPM is present and recording boot measurements. This is a basic check that the protections are turned on, not a full attestation of firmware integrity, but it is a useful signal that the foundations are in place.

2. **Record firmware versions and watch for drift (bare metal offering):** Capture component firmware versions at the start of a lease and re-check them. Unexpected changes during a lease are worth investigating.

3. **Ask how firmware is handled between tenants:** Ask whether firmware is re-flashed or re-attested before a node is reassigned. A prior tenant's changes are invisible to you, so on any reused machine this is the control that protects you.

Attack Scenarios

Supply Chain Interdiction

An adversary tampers with a GPU server before deployment and implants modified device firmware that provides covert persistence or data exfiltration. On a platform without hardware-verified boot, or where that verification is disabled or unchecked, the server is delivered, provisioned, and put into service without deep attestation, so the compromise survives into production and remains invisible for an extended period. On platforms that enforce a verified boot chain and are actually configured to do so, this implant path is far harder to land and more likely to be detected.

Remote Firmware Compromise via BMC

A customer renting a bare-metal GPU server is accidentally able to reach the BMC/IPMI interface because of faulty network segmentation. From that privileged position, they abuse firmware update mechanisms or platform controls to modify host or device firmware, creating persistence below the OS that survives reinstallation and evades standard monitoring.

Insider Firmware Tampering

A technician, contractor, or other trusted operator with maintenance access re-flashes a component, weakens integrity controls, or installs a modified image during servicing. The node returns to operation appearing healthy, but it now carries long-lived attacker-controlled logic below the operating system.

Cross-Tenant Firmware Persistence

A malicious tenant with root access to a bare-metal GPU node modifies component firmware before their lease ends. The provider wipes disks and reprovisions the server, but does not re-flash or re-attest firmware, so the next tenant may receive a compromised host. Whether the tampering succeeds in the first place depends on the platform: signed-firmware enforcement and a verified boot chain raise the bar considerably, while platforms without those controls, or with them disabled, remain exposed.

References

DMTF SPDM Specification v1.4.0. Security Protocol and Data Model standard for firmware attestation and device authentication.
https://www.dmtf.org/dsp/DSP0274

NSA ANT Catalog. Electronic Frontier Foundation, 2013. Public archive of leaked hardware and firmware implant capabilities, illustrating supply-chain and pre-boot compromise concerns.
https://www.eff.org/files/2014/01/06/20131230-appelbaum-nsa_ant_catalog.pdf

Open Compute Project Security Project. OCP resources for platform security, including secure boot, attestation, firmware signing, device identity, and secure update practices.
https://www.opencompute.org/wiki/Security

NIST SP 800-155 (Draft). BIOS Integrity Measurement Guidelines.
https://csrc.nist.gov/files/pubs/sp/800/155/ipd/docs/draft-sp800-155_dec2011.pdf

TCG TPM Library Specification. Trusted Computing Group specifications for hardware root of trust.
https://trustedcomputinggroup.org/resource/tpm-library-specification/

BlackLotus UEFI Bootkit (CVE-2022-21894). ESET, 2023. UEFI bootkit that bypassed Secure Boot on affected systems, demonstrating firmware-level persistence below the OS.
https://www.welivesecurity.com/2023/03/01/blacklotus-uefi-bootkit-myth-confirmed/

CosmicStrand UEFI Firmware Rootkit. Kaspersky, 2022. Compromised motherboard firmware persisting across OS reinstallation, with concerns about infection through hardware resale channels.
https://securelist.com/cosmicstrand-uefi-firmware-rootkit/106973/

iLOBleed, BMC Firmware Rootkit. Amnpardaz, 2021. First BMC rootkit discovered in the wild.
https://threats.amnpardaz.com/en/2021/12/28/implant-arm-ilobleed-a/

CVE-2019-6260, ASPEED BMC "Pantsdown" (CVSS 9.8). Critical BMC vulnerability enabling host firmware access through unchecked memory mappings.
https://nvd.nist.gov/vuln/detail/CVE-2019-6260

Confidential Containers (CoCo) Project. CNCF framework integrating TEEs with Kubernetes scheduling and attestation.
https://confidentialcontainers.org/

IETF RATS CoRIM (Concise Reference Integrity Manifests). IETF Remote ATtestation procedureS Working Group. Standard for vendor publication of expected attestation values used to evaluate SPDM and similar evidence
https://datatracker.ietf.org/wg/rats/about/

HPE iLO 6 User Guide, Global Component Integrity. HPE. iLO option that authenticates server components using SPDM.
https://support.hpe.com/hpesc/public/docDisplay?docId=sd00002007en_us&page=GUID-EBC183E0-5805-4BEE-AFF8-02585DC32C87.html&docLocale=en_US

FORGE-02 Network & Interconnect Vulnerabilities

Definition

Network & Interconnect Vulnerabilities occur when the high-performance fabrics used by AI clusters, such as InfiniBand, RoCE, and RDMA-based communication paths, are not properly isolated, monitored, or protected. Because these fabrics are optimized for throughput and low latency, weak controls can expose backend AI environments to discovery, lateral movement, data interception, denial of service, or workload manipulation.

Description

Modern AI infrastructure relies on high-performance interconnects to move model weights, gradients, checkpoints, and other data across GPU clusters. These environments commonly use **InfiniBand** and **RoCE**, often through RDMA, to provide low-latency communication with less dependence on the traditional kernel networking path. This improves performance, but can reduce the visibility and control available to conventional network and host security tools.

A recurring weakness is the gap between the frontend network and the backend training fabric. Frontend networks used for APIs, orchestration, monitoring, and administrative access often receive stronger access control and monitoring. Backend fabrics that carry distributed training and storage traffic may be flatter, less inspected, and weakly segmented. When fabric isolation, management controls, or workload boundaries are weak, even routine diagnostic queries from a tenant node can reveal topology, endpoints, and management details that should not be visible.

The security model depends on the transport. InfiniBand uses fabric-specific controls such as partitions, management keys, and subnet-management policies. RoCE runs over Ethernet and does not use an InfiniBand Subnet Manager, so it depends more heavily on Ethernet/IP segmentation, access controls, and device isolation. In both cases, weak defaults or poorly protected management interfaces can turn a performance fabric into a path for reconnaissance, lateral movement, or disruption.

Impact and Failure Modes

Backend traffic may lack cryptographic protection

Fabric traffic is often not encrypted by default. In native InfiniBand, keys such as **P_Key** and **M_Key** are access-control mechanisms, not cryptographic protections. In RoCE and other Ethernet-based deployments, model weights, gradients, and training data may traverse the backend network without wire-level encryption unless operators explicitly deploy protections such as MACsec or IPsec where supported.

InfiniBand partitioning may be weak or left at defaults

InfiniBand relies on P_Key for partition-level isolation. If no partition configuration is applied, OpenSM can place end ports in the broad default partition, commonly 0x7fff, increasing lateral reachability and the blast radius of a compromised node. Hardened or vendor-managed deployments may override these defaults, so the exposure depends on the actual subnet-manager configuration.

InfiniBand management-plane protection may be weak

InfiniBand uses M_Key to protect management operations on switches and adapters. In OpenSM, documented defaults such as m_key=0 and m_key_protection_level=0 leave management-key protection disabled or at the weakest level unless operators harden them explicitly. When combined with broad fabric reachability, this can expose topology, port state, and management functions that should be restricted.

RDMA memory-access credentials may be abused

RDMA uses keys such as rkeys to authorize access to registered memory regions. Research has shown that weaknesses in RDMA security mechanisms can enable unauthorized reads or writes in some environments, with limited visibility to normal host-based security tools.

Fabric management software can become a control point

Vulnerabilities in fabric management software can expose the management plane to privilege escalation, tampering, denial of service, and broader fabric compromise. For example, CVE-2024-0130 affected NVIDIA UFM and could allow improper authentication via the Ethernet management interface

RDMA isolation can be weak in containerized environments

In Kubernetes and similar platforms, shared HCA access or shared RDMA-capable devices can weaken workload isolation if operators rely only on ordinary pod-network controls and do not separately isolate RDMA devices, namespaces, or virtual functions.

Prevention And Mitigation Strategies

For providers and datacenter operators

1. **Enforce protocol-specific fabric controls, do not rely on defaults:** For InfiniBand, explicitly configure and review controls such as P_Key, M_Key, SM_Key, SA_Key, and allowed_sm_guids. For RoCE, which runs over Ethernet and has no subnet manager, apply explicit Ethernet/IP segmentation such as VLANs and ACLs. Many of these protections are weak or off in upstream defaults, so confirm they are actually set rather than assuming the fabric is isolated.

2. **Harden and isolate the fabric management plane:** Restrict access to subnet and fabric management interfaces, keep management traffic off untrusted networks, limit topology discovery and sensitive management operations from untrusted nodes, and monitor counters and management events for signs of spoofing or misconfiguration.

3. **Provide backend isolation comparable to frontend isolation:** Give the RDMA training fabric documented segmentation, workload boundaries, and tenant isolation rather than operating it as a flat internal network.

4. **Use wire-level encryption where supported and required:** For high-sensitivity workloads, deploy MACsec or IPsec on RoCE and other Ethernet-based fabrics where supported. Note that native InfiniBand keys such as P_Key and M_Key are access controls, not encryption, and in-fabric encryption on native InfiniBand is hardware-dependent and not always available, so treat backend traffic as unencrypted unless a specific protection is in place.

5. **Harden orchestration and RDMA device isolation:** Use Kubernetes NetworkPolicies for ordinary pod traffic, assign unique least-privilege service accounts per job, and regularly audit and rotate credentials to limit the blast radius of a compromise. Do not assume Kubernetes-layer controls alone secure RDMA-capable traffic, because RDMA may bypass, or be enforced outside, the ordinary pod-networking path where NetworkPolicies apply. For RDMA-capable workloads, isolate the underlying devices and network namespaces separately, for example by assigning dedicated SR-IOV virtual functions to each tenant or job rather than having multiple tenants share the same HCA interface.

For customers evaluating a provider

1. **See how much of the fabric you can enumerate:** From your node, run standard diagnostic queries such as ibstat, ibv_devinfo, saquery, sminfo, and, where permitted, ibnetdiscover. You should expect to see enough information to operate your assigned ports, but not enough to map unrelated tenants, switch topology, or management-plane details. If you can enumerate the full fabric, discover other tenants' nodes, or access management functions outside your allocation, that suggests weak fabric partitioning or management-plane protection.

2. **Check your partition membership:** Confirm which P_Key partitions your ports belong to, for example via the P_Key table exposed under /sys/class/infiniband/<device>/ports/<port>/pkeys/ or with vendor tools. If your port appears to be in the broad default partition, commonly shown as 0x7fff or full-membership 0xffff, with wide reachability rather than a dedicated tenant/job partition, isolation may have been left at defaults.

3. **For RoCE / Ethernet fabrics, scan your backend subnet:** Since RoCE rides on Ethernet, ordinary IP tools apply: look at what's in your ARP table, check your subnet mask (is the backend a flat /16 with everyone on it?), and see whether you can reach other hosts' IPs and ports on the storage or training network. Reaching other tenants' addresses indicates missing VLAN/segmentation.

4. **Look at port counters for noisy-neighbor signs:** Your own port error and congestion counters, for example via perfquery or /sys/class/infiniband/<device>/ports/<port>/counters/, can show link errors, congestion, retransmits, drops, or other symptoms that may point to shared-fabric contention. These counters are useful evidence, but they should be interpreted against a baseline and your workload profile rather than treated as proof of co-tenant interference.

Attack Scenarios

Fabric Management Abuse
An attacker gains a foothold on a tenant node through a compromised container, workload, or account. From that node, they use fabric diagnostic and management queries to map the backend interconnect, discover topology, identify node addresses, and inspect port or switch state. If the InfiniBand fabric is weakly partitioned, such as broad use of the default P_Key 0x7fff, and management protections such as M_Key are unset or weakly configured, the attacker may gain visibility or influence over fabric behavior outside their allocation. In a hardened environment, tenant nodes should not be able to enumerate unrelated topology or perform management actions beyond their assigned fabric scope.

RDMA Memory Exposure
An attacker abuses weaknesses in RDMA memory-access protections to obtain or misuse credentials such as rkeys for registered memory regions. They then issue one-sided RDMA reads or writes against memory that should not be accessible to them. Because one-sided RDMA operations can complete without active involvement from the remote CPU, the attacker may extract training data, checkpoints, gradients, or model-related buffers with limited visibility to normal host-based security tools.

RDMA Noisy Neighbor
In a shared AI environment, a malicious tenant abuses RDMA-capable devices, shared HCAs, or interconnect resources to consume bandwidth, trigger congestion, or exhaust RNIC resources. Neighboring distributed training jobs may experience degraded throughput, higher latency, failed synchronization, or stalled execution. The attacker does not need to read or modify another tenant's data. The failure is that one tenant can materially degrade another tenant's workload through shared low-level fabric resources.

References

NVIDIA InfiniBand Security Overview and Guidelines. NVIDIA. Vendor guidance on IB fabric security controls including P_Key, M_Key, and subnet management.
https://docs.nvidia.com/networking/display/nvidiainfinibandsecurityoverviewandguidelines/security+in+infiniband

OpenSM Partition Configuration. linux-rdma. P_Key partition configuration reference for InfiniBand subnet managers.
https://github.com/linux-rdma/opensm/blob/master/doc/partition-config.txt

NVIDIA MLNX_OFED, Kubernetes with Shared HCA. NVIDIA. RDMA device sharing and isolation in Kubernetes environments.
https://docs.nvidia.com/networking/display/mlnxofedv23106161lts/Kubernetes-with-Shared-HCA

Distributed Training Attack Surface. Red Teams AI. Security analysis of distributed training communication and orchestration.
https://redteams.ai/topics/training-pipeline/advanced/distributed-training

opensm(8) man page, M_Key Configuration. Debian. OpenSM management key defaults and hardening options.
https://manpages.debian.org/testing/opensm/opensm.8.en.html#MKEY_CONFIGURATION

OpenSM Source, osm_base.h. linux-rdma. Default M_Key and protection level constants in source code.
https://github.com/linux-rdma/opensm/blob/master/include/opensm/osm_base.h

Set up InfiniBand on HPC VMs. Microsoft Azure. InfiniBand configuration for cloud HPC and AI workloads.
https://learn.microsoft.com/en-us/azure/virtual-machines/setup-infiniband

CVE-2024-0130. NVIDIA UFM improper authentication via Ethernet management interface.
https://nvd.nist.gov/vuln/detail/CVE-2024-0130

CVE-2024-0101. Mellanox OS ipfilter bypass.
https://nvd.nist.gov/vuln/detail/CVE-2024-0101

ReDMArk: Bypassing RDMA Security Mechanisms. Rothenberger et al., USENIX Security 2021. Demonstrates weaknesses in RDMA memory-access credentials that enable unauthorized remote reads and writes.
https://www.usenix.org/system/files/sec21-rothenberger.pdf

NeVerMore: Exploiting RDMA Mistakes in NVMe-oF Storage Applications. Taranov et al., arXiv 2022. RDMA vulnerabilities in storage applications.
https://arxiv.org/abs/2202.08080

Noisy Neighbor: Exploiting RDMA for Resource Exhaustion in Containerized Clouds. SecAssure 2025. RDMA-based denial-of-service in shared container environments.
https://arxiv.org/abs/2510.12629

RFC 4391, Transmission of IP over InfiniBand. IETF. Section 13 covers security considerations for IPoIB deployments.
https://www.rfc-editor.org/rfc/rfc4391

NIST SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations.https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final
Distributed Training Attack Surface. Red Teams AI. Security analysis of distributed training communication and orchestration. redteams

FORGE 03: Unsafe Multi-Tenant Isolation And Resource Reuse

Definition

Unsafe Multi-Tenant Isolation and Resource Reuse occurs when shared, partitioned, or reassigned AI infrastructure fails to maintain a reliable boundary between tenants, workloads, or execution environments. This can allow one tenant to escape its assigned boundary, observe residual state, discover neighboring workloads, affect the host, or degrade other tenants through shared compute, storage, orchestration, GPU, or interconnect resources.

Description

AI infrastructure often relies on shared, partitioned, or rapidly reassigned compute resources to maximize scarce accelerator capacity. This creates tenant-isolation problems because workloads may share more than an application API. They may also share physical GPUs, bare-metal hosts, local storage, host kernels, driver stacks, container runtimes, orchestration layers, DNS and service discovery systems, RDMA fabrics, and high-speed interconnects. This risk focuses on failures in how that shared or reassigned capacity is isolated, cleaned, and reused across tenants, workloads, or execution environments. The risk arises when isolation at one layer is mistaken for isolation across the full stack. A workload may be separated by a Kubernetes namespace, container runtime, MIG partition, vGPU profile, or dedicated GPU assignment, while still sharing underlying host, driver, storage, network, or orchestration layers. MIG, vGPU, sandboxed runtimes, and TEE-based confidential computing can strengthen specific parts of the boundary, but none provides complete tenant isolation on its own. Isolation can fail actively or passively. Active failures include container or VM escapes, GPU runtime compromise, weak Kubernetes RBAC, unsafe device access, or abuse of shared RDMA and interconnect resources. Passive failures are quieter: residual VRAM contents, caches, local disk data, credentials, logs, or orchestration metadata may remain after a resource is reused. This is not limited to live co-tenancy. A bare-metal GPU server may be leased to one customer, returned to the provider pool, and reassigned to another. If the reset process only reloads the operating system image, it may leave local disks, device state, firmware settings, driver state, cached artifacts, credentials, logs, or temporary workload data untouched. Without deliberate cleanup and verification across the full stack, the next tenant may inherit residual data or state from the previous environment.

Impact and Failure Modes

Host or cluster compromise through GPU runtime integration

GPU workloads often require privileged runtime components, device mounts, container hooks, drivers, and access to accelerator interfaces. Weaknesses in this stack can allow a crafted container image or tenant workload to access the host filesystem, runtime sockets, credentials, mounted paths, GPU device interfaces, or neighboring workloads on the same node. One example is CVE-2024-0132, an NVIDIA Container Toolkit container escape.

GPU virtualization failures can cross the guest-host boundary

In virtualized GPU environments, a tenant VM may affect the host through the GPU virtualization or management layer. One example is CVE-2024-0099, an NVIDIA vGPU issue where a guest OS could trigger a buffer overrun in the host Virtual GPU Manager.

Weak orchestration and service discovery boundaries

Kubernetes namespaces do not provide strong tenant isolation by themselves. Reused service accounts, weak RBAC, shared node pools, unsafe runtime classes, broad device assignment, or shared DNS and service discovery can allow one tenant to discover or reach resources associated with another tenant.

Host or cluster compromise through GPU runtime integration

GPU workloads often require privileged runtime components, device mounts, container hooks, drivers, and access to accelerator interfaces. Weaknesses in this stack can allow a crafted container image or tenant workload to access the host filesystem, runtime sockets, credentials, mounted paths, GPU device interfaces, or neighboring workloads on the same node. One example is CVE-2024-0132, an NVIDIA Container Toolkit container escape.

Cross-tenant interference through shared RDMA or interconnect resources:

A tenant may not need to compromise another tenant directly to affect their workload. In shared AI clusters, abuse or overconsumption of RDMA, NIC, or interconnect resources can degrade neighboring jobs, increase latency, reduce bandwidth, or stall distributed training synchronization.

GPU state and side channels may leak across contexts

Research such as Whispering Pixels and NVBleed has shown that GPU state, memory behavior, and multi-GPU interconnect activity can leak information across execution contexts under specific conditions. These are research findings rather than routine production attacks, but they remain an open problem where mutually untrusted workloads share GPUs or interconnects.

Cross-tenant interference through shared RDMA or interconnect resources

A tenant may not need to compromise another tenant directly to affect their workload. Abuse or overconsumption of RDMA, NIC, or interconnect resources can degrade neighboring jobs, increase latency, reduce bandwidth, or stall distributed training synchronization.

Prevention And Mitigation Strategies

For providers and datacenter operators

1. **Harden GPU runtime boundaries:** Keep GPU drivers, NVIDIA Container Toolkit, device plugins, and container runtimes patched. Restrict privileged containers, hostPath mounts, broad device access, unsafe container hooks, and unnecessary access to accelerator interfaces. Treat GPU runtime integration as part of the tenant-isolation boundary, not as a trusted implementation detail.

2. **Treat bare-metal reassignment as a security boundary:** Before a bare-metal GPU server is reassigned to another tenant, reset and revalidate the full stack, not only the operating system. This includes GPU and device state, MIG configuration, local storage, cached artifacts, orchestration metadata, host identifiers, credentials, and tenant-specific logs, and it includes confirming firmware and driver state is known-good rather than inherited from the prior tenant.

3. **Separate high-risk workloads by tenancy model:** Do not treat containers, Kubernetes namespaces, vGPU, MIG, dedicated GPUs, dedicated hosts, and dedicated bare-metal clusters as equivalent isolation models. Use stronger boundaries for untrusted tenants, sensitive training workloads, regulated data, or customers requiring no co-tenancy.

4. **Harden orchestration boundaries:** Use per-tenant namespaces, least-privilege service accounts, strong RBAC, admission controls, dedicated runtime classes where appropriate, and separate node pools for higher-trust and lower-trust workloads. Avoid sharing service accounts, runtime permissions, device plugins, or management components across tenants unless the isolation model has been explicitly reviewed.

5. **Protect service discovery and DNS boundaries:** Ensure that tenants cannot enumerate other tenants' services, records, hostnames, internal architecture, or infrastructure metadata through reverse DNS, shared DNS zones, Kubernetes service discovery, or predictable naming schemes. Restrict cross-tenant DNS visibility, avoid tenant-identifying names in shared records, and test service discovery exposure from a normal tenant position.

6. **Isolate shared model-loading paths:** Treat model loaders, inference services, notebook runtimes, and artifact-processing pipelines as tenant-boundary components when they process customer-supplied files. Do not load untrusted models, checkpoints, or serialized artifacts inside shared or highly privileged services unless the loading path is sandboxed and isolated.

7. **Offer stronger isolation and attestation options for high-sensitivity workloads:** Where the platform supports it, provide dedicated hosts, dedicated clusters, or hardware-backed confidential computing / TEE-based isolation with remote attestation for customers that require stronger protection from the host, hypervisor, or provider-managed layer. Clearly document whether capabilities such as NVIDIA GPU Confidential Computing on H100/H200 are available, what they protect, what remains trusted, and what evidence customers can verify before workload execution.

8. **Keep the GPU virtualization stack patched:** Quickly patch host-side GPU management components, drivers, and device plugins in vGPU, MIG, or other GPU-sharing setups (for example CVE-2024-0099).

9. **Reduce cross-context leakage risk:** Clear or reinitialize GPU memory, including VRAM, as well as GPU contexts, runtime state, MIG or vGPU configuration, and other residual accelerator state before a GPU, MIG slice, vGPU, or bare-metal node is reassigned to another tenant. Restrict low-level diagnostic, profiling, and debug interfaces unless they are required and safely scoped to the tenant.

10. **Offer Trusted Execution Environment and stronger isolation options:** Where the platform supports it, provide hardware-backed confidential computing / Trusted Execution Environment (TEE)-based isolation with remote attestation for workloads that require stronger protection from the host, hypervisor, or provider-managed layer. Clearly document whether NVIDIA GPU Confidential Computing or similar capabilities are available, what they protect, and what attestation evidence customers can verify before releasing secrets to the workload.

11. **Control shared RDMA and interconnect resource abuse:** Monitor fabric utilization, congestion, error counters, packet drops, abnormal bandwidth patterns, and synchronization delays. Use fabric partitioning, tenant-level limits where available, congestion monitoring, and isolation of high-sensitivity workloads. Treat RDMA, NIC, and interconnect resources as part of the tenant boundary in distributed AI environments.

For customers evaluating a provider

How much you can check depends on the offering. On a dedicated or bare-metal node you can inspect the state you are handed. On MIG, vGPU, or fractional offerings you see less and rely more on the provider.

What a customer can often verify directly:

1. **Check the GPU is clean when you receive it:** At the start of a lease, confirm that no unexpected GPU processes, compute contexts, or unexplained memory usage are visible, for example with nvidia-smi. Some memory may be reserved by the driver or platform, so this is a sanity check rather than proof of secure scrubbing. On a shared GPU, seeing processes or contexts outside your own work is a warning sign. On a MIG instance, vGPU, or VM, you should see only the resources and boundary assigned to you.

2. **Check local scratch storage and caches for residual data:** Inspect any local NVMe or SSD scratch disks attached to the node, along with common cache locations, for leftover datasets, checkpoints, logs, containers, credentials, or model files from a previous tenant. Finding any such data is direct evidence that tenant reset and cleanup are incomplete.

3. **Confirm which isolation model you actually have:** Verify whether you have a hardware-partitioned MIG instance, a vGPU slice, a dedicated GPU, a dedicated host, or a full bare-metal node. These models provide different visibility, performance, and isolation properties and should not be treated as equivalent.

Attack Scenarios

Host Access Through GPU Runtime Escape
A tenant submits a crafted GPU container image to a managed AI environment. The image abuses CVE-2024-0132, a vulnerability in NVIDIA Container Toolkit, or a similar weakness in the GPU runtime integration path. The tenant workload gains access to host files or GPU device interfaces outside the assigned workload boundary. From the host, the attacker may reach credentials, mounted paths, runtime sockets, or neighboring workloads on the same node.

Cross-Tenant Service Discovery Through Reverse DNS
A tenant receives access to an environment that is intended to be isolated from other tenants. From that tenant position, the attacker performs reverse DNS lookups against shared infrastructure address space and discovers Kubernetes service names or records associated with other tenants. The attacker does not need initial access to those tenants' workloads. The failure is that shared service discovery or DNS metadata exposes cross-tenant information that should not be visible from a tenant environment. This can reveal tenant identities, service names, internal architecture, workload purpose, or possible follow-on targets.

Cross-Tenant Interference Through Shared RDMA Resources
A tenant runs a workload that heavily consumes shared RDMA, NIC, or interconnect resources in a multi-tenant AI cluster. Neighboring distributed training jobs experience bandwidth degradation, latency inflation, or stalled synchronization. The attacker does not need to read another tenant's data. The isolation failure is that one tenant can materially degrade another tenant's execution environment through shared low-level resources.

Residual State Exposure After Bare-Metal Reassignment
A bare-metal GPU server is leased to one tenant, returned to the provider pool, and then assigned to another tenant. The provider reinstalls the operating system but does not fully reset local storage, GPU state, cached artifacts, host identifiers, runtime configuration, or orchestration metadata. The next tenant discovers traces of the previous environment, exposing information that should have been removed before reassignment.

References

Wiz Research, Critical NVIDIA AI Vulnerability, 2024. Deep dive on NVIDIA Container Toolkit escape, CVE-2024-0132.
https://www.wiz.io/blog/wiz-research-critical-nvidia-ai-vulnerability

Wiz Research, NVIDIA AI Vulnerability Deep Dive, CVE-2024-0132, 2024.
https://www.wiz.io/blog/nvidia-ai-vulnerability-deep-dive-cve-2024-0132

NVIDIA vGPU software issue referenced in AIS 03, CVE-2024-0099, guest OS to host Virtual GPU Manager buffer overrun.
Whispering Pixels, Exploiting Uninitialized Register Accesses in Modern GPUs. Cross context GPU leakage across workloads.
https://arxiv.org/abs/2401.08881

NVIDIA Confidential Containers Reference Architecture. NVIDIA. Describes NVIDIA GPU Confidential Computing support for running GPU workloads inside a hardware-enforced Trusted Execution Environment on Kubernetes.
https://docs.nvidia.com/datacenter/cloud-native/confidential-containers/latest/overview.html

NVBleed, Covert and Side Channel Attacks on NVIDIA Multi GPU Interconnect. Side channel attacks over NVLink interconnects.
https://arxiv.org/abs/2503.17847

Kubernetes Multi tenancy Documentation. Guidance on namespace isolation, RBAC, and workload separation.
https://kubernetes.io/docs/concepts/security/multi-tenancy/

ReDMArk, Bypassing RDMA Security Mechanisms. Rothenberger et al, USENIX Security 2021. Demonstrates RDMA credential weaknesses enabling unauthorized memory access.
Confidential Containers (CoCo) Project. CNCF project for running cloud-native workloads inside hardware-backed Trusted Execution Environments with attestation support.
https://confidentialcontainers.org/

FORGE-04 Insecure Out-Of-Band Management Plane

Definition

Insecure Out-of-Band Management Plane occurs when the systems used to manage physical infrastructure outside the host operating system are exposed, weakly authenticated, or poorly monitored. This plane includes BMCs, IPMI and Redfish interfaces, dedicated management networks, jump hosts, fleet-management and firmware-update tools, management switches.

Description

In AI data centers, out-of-band (OOB) management allows administrators to control physical infrastructure independently of the host operating system. The core of this system is often the Baseboard Management Controller (BMC), a separate management processor on each server that handles remote functions such as power control, hardware monitoring, firmware update workflows, remote console access, and virtual media.

The management plane extends beyond individual BMCs. It can include:

- **The Dedicated Management Network:** The physical and virtual infrastructure connecting OOB endpoints, such as management VLANs, jump hosts, VPNs, and management switches.
- **Management Protocols:** Interfaces used to access these systems, such as IPMI, Redfish APIs, SSH, web consoles, and vendor-specific management interfaces.
- **Centralized Fleet Management Tools:** Platforms that store credentials, inventory systems, and automation workflows used to manage large numbers of BMCs and servers at scale (e.g., Dell OpenManage Enterprise).
- Power Distribution Units (PDUs): Network-connected smart PDUs and related systems that can remotely cut, cycle, or sequence power to racks and devices.
- **Firmware Update Channels:** The mechanisms used to inventory, distribute, validate, and install firmware for platform components such as BIOS/UEFI, BMCs, NICs, GPUs, and storage controllers, for example through fleet-management tools such as Dell OpenManage Enterprise.

The security posture of this layer varies significantly by OEM, platform generation, and configuration. Mature server platforms may include hardware roots of trust, signed firmware enforcement, integrity validation, lockdown modes, and recovery mechanisms that reduce the risk of low-level tampering when enabled and verified.

The residual risk is highest where OOB interfaces are reachable from shared or weakly segmented networks, credentials are weak or overprivileged, firmware update paths are insufficiently protected, or management activity is poorly monitored. In those environments, management-plane compromise can give an attacker below-OS capabilities such as remote power control, virtual media mounting, firmware update abuse, OS reinstallation, or persistent access that may survive ordinary host remediation.

If centralized fleet-management tooling is compromised, the blast radius can extend from a single node to large parts of the fleet.

Impact and Failure Modes

Default or weak BMC credentials

Many BMC platforms have historically shipped with default or weak credentials. Newer platforms increasingly ship with unique per-device factory passwords or forced first-login changes, which reduces this exposure, so the risk now concentrates on older deployments and devices left at factory defaults. Even per-device sticker passwords may have lower real entropy than expected, leaving them vulnerable to offline cracking unless rotated.

Legacy or weakly protected IPMI sessions

IPMI 2.0 includes insecure options such as Cipher Suite 0 and other suites with no integrity or confidentiality. When enabled, these weaken or eliminate session protection and can expose management traffic to interception or abuse.

Credential reuse across large fleets

Reused BMC or fleet-management credentials can turn one compromised account into broad management-plane access. Historical IPMI research also showed cases where credential material or password hashes could be exposed in ways that enabled offline cracking or recovery.

Enabled host-to-BMC interfaces

In-band interfaces such as KCS, USB/RNDIS, pass-through, or Redfish Host Interface can allow host-level software to interact with the management controller. If those interfaces are exposed, vulnerable, or misconfigured, a host compromise may become a path into the management plane.

Smart PDU compromise enabling physical denial of service

An attacker with access to PDU management interfaces can power-cycle racks or entire rows, halting multi-day training jobs and potentially corrupting in-flight checkpoints.

Critical Redfish and BMC API vulnerabilities

Vendor Redfish and IPMI implementations have repeatedly exposed remote authentication bypass and code execution flaws. Recent examples include CVE-2023-34329, CVE-2023-34330, and CVE-2024- 54085 in AMI MegaRAC-derived platforms, as well as earlier critical flaws such as CVE-2017-12542 in HPE iLO4.

Poor patching and limited visibility

Slow BMC firmware patch cycles, incomplete inventory, and weak audit collection can leave known vulnerabilities unremediated for long periods and make malicious management-plane activity difficult to detect.

Direct customer BMC exposure in hosted environments:

Management interfaces left reachable from tenant networks or the public internet due to misconfigured routing or weak network segmentation, without isolation or monitoring in place to detect unauthorized access.

Adjacent management-network devices can become control points

Smart PDUs, console servers, KVM systems, management switches, and SNMP-managed devices can expose inventory, configuration details, console access, or physical control if they use weak credentials, legacy protocols, or shared SNMP community strings.

Prevention And Mitigation Strategies

For providers and datacenter operators

1. **Segment and monitor the management network, not just its perimeter:** Place BMCs and other out-of-band endpoints on a dedicated management network or VLAN with no routing from workload, storage, public-facing, or corporate networks, and never expose them directly to the internet. Treat the network as a monitored trust zone: limit lateral movement between management devices and alert on unexpected traffic, so one compromised endpoint cannot freely reach every other.

2. **Replace factory credentials and harden authentication:** Replace factory-default and per-device sticker passwords before deployment, since sticker passwords are often shorter and drawn from a narrower character set than they appear and can be cracked offline. Store replacements in an approved secrets-management system, enforce strong authentication, prefer modern encrypted interfaces, and disable insecure IPMI options such as Cipher Suite 0 where possible.

3. **Restrict who can reach management interfaces:** Limit Redfish, IPMI, SSH, web UI, and vendor-specific interfaces to authorized jump hosts or management systems, using ACLs, IP filtering, VPN access, and MFA where supported.

4. **Disable or minimize host-to-BMC interfaces:** Disable in-band interfaces such as KCS, USB networking, pass-through, or Redfish Host Interface unless operationally required. Where they must remain enabled, minimize privileges and restrict access paths tightly.

5. **Patch and verify BMC firmware:** Maintain an inventory of firmware versions and apply vendor updates promptly, using vendor-signed images verified before writing to flash, and disable unsafe update methods. Keep recovery procedures for cases that may require physical re-flashing. Where attestation is available, use it to verify BMC state, but note that self-attestation has limited coverage today and that fleet inventory often relies on version numbers self-reported by the BMC itself.

6. **Collect independent logging and monitoring:** Log logins, power actions, firmware updates, virtual media mounts, console sessions, and account changes to a centralized system that does not depend on the BMC itself.

7. **Treat fleet and OOB management tools as crown-jewel systems:** Centralized BMC and fleet managers such as Dell OpenManage or Redfish aggregators hold credentials to the entire management plane, so a single compromise can reach every node. Restrict them to dedicated admin workstations or bastions, require MFA and least-privilege roles, log all actions, and use unique credentials per device or trust domain so credential reuse cannot turn one foothold into fleet-wide control.

8. **Harden other management-network devices:** Smart PDUs, serial console servers, and management switches often ship with default or shared credentials and legacy SNMP, and a PDU that can cut power to a rack is an availability risk on its own. Change defaults, replace SNMPv1/v2c community strings with SNMPv3, and apply the same segmentation and access controls used for BMCs.

For customers evaluating a provider

1. **Test what management systems you can reach:** From your node and its network, confirm you cannot reach your BMC/IPMI interface, the management subnet, or other management devices such as PDUs, console servers, or switch management ports. Reaching any of these, including another tenant's BMC, is a segmentation failure you have found directly.

2. **Check for in-band host-to-BMC paths:** On bare metal, check whether interfaces such as KCS, USB/RNDIS, or the Redfish Host Interface are exposed from your OS, since these are the path used to pivot from a host compromise into the management plane.

Attack Scenarios

Cluster-Wide Management Plane Takeover

An attacker compromises an exposed or weakly protected BMC using a vulnerability or default credentials. A single BMC compromise does not automatically expose the fleet, but it can become a wider management-plane compromise when credentials are reused, the management network is flat, or the same vulnerable stack is deployed across many servers. The attacker may then enumerate reachable BMCs, reuse credentials, repeat the same exploit, or pivot toward centralized fleet-management tooling.

Persistent Firmware Implant via BMC

An attacker who gains BMC access exploits a vulnerable update path or modifies BMC firmware directly to implant persistent malicious code. The implant survives host OS reinstallation, disk replacement, and ordinary node reprovisioning because it resides in the BMC's own flash storage. In more severe cases, the compromise may also extend into server SPI flash, affecting broader platform trust and making recovery dependent on physical reflashing. Real-world reporting on iLOBleed demonstrated that BMC firmware implants have appeared outside the lab.

Host-to-BMC Pivot Through In-Band Interfaces

An attacker first gains root access on a host through an application, container, or OS compromise. The environment leaves KCS, USB/RNDIS, pass-through, or Redfish Host Interface enabled between the host and the BMC. The attacker uses that path to interact with the management controller, create new accounts, change settings, or pursue firmware-level persistence. In some architectures, local IPMI access means this pivot may not require separate BMC credentials. The result is that an ordinary host compromise is upgraded into a lower-level management-plane compromise that survives OS rebuilding and sits outside normal host-based security visibility.

References

Breaking BMC: The Forgotten Key to the Kingdom. DEF CON. BMC attack surface overview and exploitation techniques.
https://forum.defcon.org/node/245714

Vulnerable Firmware in the Supply Chain of Enterprise Servers. Eclypsium. Firmware supply-chain risk across server platforms.
https://eclypsium.com/wp-content/uploads/Vulnerable-Firmware-in-the-Supply-Chain.pdf

Analyzing Baseboard Management Controllers to Secure Data Center Infrastructure. NVIDIA Technical Blog. BMC security analysis for AI data centers.
https://developer.nvidia.com/blog/?p=100850

NVIDIA DGX Security Guidance, Securing the BMC Port. NVIDIA. DGX-specific BMC hardening recommendations.
https://docs.nvidia.com/dgx/dgxh100-user-guide/security.html

Dell iDRAC Security Best Practices. Dell. Vendor BMC hardening guidance for iDRAC9.
https://www.dell.com/support/manuals/en-us/idrac9-lifecycle-controller-v7.x-series/idrac9_scg_tta/best-practices?guid=guid-a7415277-ace9-40bd-9017-467a8c133165&lang=en-us

DMTF Redfish Host Interface Specification (DSP0270). DMTF. Standard for host-to-BMC in-band communication.
https://www.dmtf.org/dsp/DSP0270

A Penetration Tester's Guide to IPMI and BMCs. Rapid7, 2013. IPMI attack techniques, default credentials, and enumeration.
https://www.rapid7.com/blog/post/2013/07/02/a-penetration-testers-guide-to-ipmi/

IPMI: Freight Train to Hell. Dan Farmer, 2013. Foundational IPMI security research covering protocol weaknesses and credential exposure.
http://fish2.com/ipmi/itrain.pdf

OpenBMC Project. Linux Foundation. Open-source BMC firmware stack.
https://openbmc.org/

CVE-2023-34329. AMI MegaRAC Redfish authentication bypass.
https://nvd.nist.gov/vuln/detail/CVE-2023-34329

CVE-2023-34330. AMI MegaRAC Redfish code execution via Host Interface.
https://nvd.nist.gov/vuln/detail/CVE-2023-34330

CVE-2024-54085. AMI MegaRAC critical BMC vulnerability.
https://nvd.nist.gov/vuln/detail/CVE-2024-54085

CVE-2017-12542. HPE iLO4 authentication bypass (CVSS 10.0).
https://nvd.nist.gov/vuln/detail/CVE-2017-12542

BMC&C: Lights Out Forever. Eclypsium. BMC-based persistent access and command-and-control techniques.
https://eclypsium.com/research/bmcc-lights-out-forever/

The iLOBleed Implant. Eclypsium, 2021. First BMC firmware rootkit discovered in the wild.
https://eclypsium.com/blog/the-ilobleed-implant-lights-out-management-like-you-wouldnt-believe/

Turning your BMC into a revolving door. Airbus Security Lab, ZeroNights 2018. iLO exploitation techniques including firmware analysis and persistent access.
https://airbus-seclab.github.io/ilo/ZERONIGHTS2018-Slides-EN-Turning_your_BMC_into_a_revolving_door-perigaud-gazet-czarny.pdf

BMC Attack Surface via Host-to-BMC Interfaces. ScienceDirect, 2020. Host-to-BMC DMA and memory access risks.
https://www.sciencedirect.com/science/article/pii/S2666281720300147

FORGE-05 Supply Chain Compromise In AI Frameworks

Definition

AI Infrastructure Supply Chain Compromise refers to the introduction of compromised, tampered, or untrusted components into AI environments through upstream supply-chain relationships. This includes hardware, firmware, software artifacts, images, and infrastructure templates that cross from vendors, integrators, registries, public sources, or internal mirrors into the operator's trusted environment.

Description

The AI infrastructure supply chain spans two lanes. The physical lane includes servers, accelerators, NICs, storage devices, BMCs, and other components that pass through manufacturers, integrators, logistics providers, and refurbishment paths before reaching production. A system may arrive with firmware, management configuration, or node images prepared by a third party, and the operator may not have independently verified that state before deployment.
The digital lane includes container images, GPU drivers and runtimes, Python packages, Kubernetes operators, model-serving components, infrastructure-as-code templates, and model artifacts pulled from vendor catalogs, public registries, open-source repositories, or internal mirrors. These components may be legitimate, maliciously tampered with, outdated, misconfigured, or pulled from an unexpected source.

Both lanes share the same core problem: components are often trusted because they arrive through an expected path, not because their integrity, provenance, configuration, and update history were independently verified. This dependency can continue throughout the product lifecycle, as operators rely on vendors to update cryptographic mechanisms when standards and threats evolve. Much of this risk is addressable with widely available controls, including signed images and artifacts, build provenance, dependency pinning, SBOMs, registry scanning, controlled internal mirrors, and safer model formats such as safetensors where applicable. The exposure concentrates where these controls are not adopted, where artifacts are pulled directly from public sources into production, or where internal mirrors are trusted without verifying integrity, provenance, and update history.

See also: AIS-10 Vendor Embargo Gaps & Patch Velocity Failures for container runtime and driver patching risks.

Impact and Failure Modes

Unverified third-party staging or integration

A system integrator, reseller, or logistics partner preloads firmware, management credentials, BIOS settings, or node images before delivery. If the provider accepts the systems without independent verification, it may inherit whatever configuration, vulnerability, or implant was introduced upstream.

Unsafe model checkpoint loading

Some model-loading paths, especially pickle-based formats, can execute code during deserialization. A malicious or tampered model artifact can therefore turn the act of loading a model into code execution on training, evaluation, or inference nodes.

Poisoned AI infrastructure dependency or container image

A compromised CUDA or NCCL dependency, GPU operator, device plugin, model-serving image, or privileged base image can introduce credential theft, backdoors, or cluster-level access into GPU nodes and Kubernetes environments that automatically pull or update trusted infrastructure components.

Tampered internal mirror or registry

An attacker who compromises an internal package index, container registry, model registry, or artifact repository can replace trusted components with malicious versions. Because production systems often trust internal mirrors more than public sources, the compromise can spread quickly across workloads, nodes, or clusters.

Compromised node image or provisioning template

A malicious or accidental change to a golden image, bootstrap script, Kubernetes manifest, Terraform module, or build pipeline can propagate to every GPU worker provisioned from that artifact.

Lack of cryptographic agility in long-lived systems

An AI or HPC platform may remain in service for many years while depending on vendors to update firmware signing, key exchange, digital signatures, and authentication mechanisms. If the hardware or firmware cannot support newer algorithms, including future post-quantum requirements, the operator may be forced to continue using outdated cryptography or replace the platform.

Prevention and Mitigation Strategies

For providers and datacenter operators

1. **Verify before trust at every handoff: I**ndependently verify firmware versions, BIOS and management settings, installed images, and component inventory when receiving systems from integrators, resellers, logistics partners, manufacturers, or refurbishment paths. Re-establish trust after any third-party staging, repair, or return rather than accepting the delivered state.

2. **Control and restrict artifact sources:** Mirror approved packages, container images, GPU drivers, model artifacts, Helm charts, and infrastructure-as-code templates into controlled internal registries. Do not allow production systems to pull directly from unreviewed public registries or repositories. Pin images by immutable digest, lock dependency versions, and require signatures, checksums, or build provenance for artifacts in privileged infrastructure paths.

3. **Treat privileged AI infrastructure components as high-risk:** Apply extra scrutiny to components that run with cluster or node privilege, such as GPU operators, device plugins, CUDA/NCCL dependencies, drivers, and privileged base or model-serving images. A compromise in these components can reach GPU nodes or the Kubernetes control plane directly, so they should come only from verified sources and be pinned by digest where possible.

4. **Restrict build and provisioning systems:** Limit cloud credentials, registry write access, Kubernetes privileges, and outbound connectivity for image builders, CI/CD pipelines, and fleet automation. A compromised build step should not be able to exfiltrate secrets, overwrite trusted artifacts, or fetch follow-on payloads.

5. **Maintain inventory and prepare for recall:** Track SBOMs, hardware bills of materials, image digests, firmware versions, dependency versions, and deployment locations so a compromised artifact or component can be identified, blocked, and removed quickly across the fleet.

For customers and teams running their own pipelines

A meaningful share of supply-chain risk can be reduced in your own pipeline, while the physical hardware and provider-managed base-image lane require asking the provider for evidence.

1. **Pin and verify what you pull:** Pin container images by digest, lock dependency versions, and verify signatures where available so upstream changes cannot silently enter your environment.

2. **Prefer safer model-loading paths:** Load models from safer formats such as safetensors rather than pickle-based checkpoints where possible, and scan model artifacts before loading them into privileged training, evaluation, or inference environments.

3. **Mirror and scan instead of pulling directly from public sources:** Pull packages, images, and models through a controlled internal mirror that scans artifacts before promotion to production.

4. **Limit what build and training jobs can reach:** Scope credentials and outbound network access for CI/CD and training jobs so a compromised dependency cannot exfiltrate secrets or fetch a follow-on payload.

Attack Scenarios

Typosquatted ML Package

An attacker publishes a malicious package with a name similar to a popular ML dependency, or compromises a legitimate upstream package. A data scientist installs or updates it during environment setup. The package executes during installation or import and steals credentials, model artifacts, or training data, or establishes a foothold for later access.

Malicious Model Artifact in a GPU Pipeline

A team downloads a popular pretrained model from a public hub and loads it directly into a privileged training or inference environment. Incidents involving malicious pickle-based checkpoints on Hugging Face showed that model loading can become code execution when unsafe serialization formats are used. Even when the format itself is non-executable, a poisoned model may still pass basic validation and later produce attacker-controlled behavior on selected inputs.

Compromised System Integrator Baseline

A provider receives GPU servers from a third-party integrator that assembled, racked, and imaged the systems before delivery. The servers arrive with an operating system already installed, along with firmware updates and management configuration applied during staging. The provider verifies that the systems boot and pass health checks, but does not independently verify what was included in the pre-installed OS image. The image contains a persistent backdoor that survives the provider's standard onboarding process.

Compromised Firmware Signing Key

An attacker obtains a server vendor's firmware signing key. They produce a malicious BMC firmware image that passes the provider's signature verification because it is signed with the legitimate key. The attacker uploads it to the vendor's firmware support portal. The provider downloads and deploys it fleet-wide as a routine update. Once installed, it provides persistent remote access to the management plane on every node that accepted the update.

References

NIST SP 800-218, Secure Software Development Framework. NIST. Secure SDLC practices for software producers.
https://csrc.nist.gov/pubs/sp/800/218/final

OpenSSF Scorecard. Open Source Security Foundation. Automated security health checks for open-source projects.
https://github.com/ossf/scorecard

PyTorch torch.load() Documentation. PyTorch. Documents unsafe deserialization risks in pickle-based model loading.
https://docs.pytorch.org/docs/stable/generated/torch.load.html

Pickle Scanning and Pickle Security. Hugging Face. Model artifact malware scanning on the Hub.
https://huggingface.co/docs/hub/security-pickle

Hugging Face Hub Security. Hugging Face. Platform-level model and artifact security controls.
https://huggingface.co/docs/hub/security

Safetensors, Secure Model Weight Format. PyTorch / Hugging Face. Safe alternative to pickle-based model serialization.
https://pytorch.org/projects/safetensors/

NVIDIA NGC, Signed Container Images. NVIDIA. Verified GPU container image provenance.
https://docs.nvidia.com/ngc/gpu-cloud/ngc-catalog-user-guide/index.html

Chainguard Images Documentation. Chainguard. Minimal, hardened container base images.
https://edu.chainguard.dev/chainguard/chainguard-images/

Exporting a Software Bill of Materials. GitHub. SBOM generation for dependency tracking.
https://docs.github.com/en/code-security/supply-chain-security/understanding-your-software-supply-chain/exporting-a-software-bill-of-materials-for-your-repository

Data Scientists Targeted by Malicious Hugging Face ML Models with Silent Backdoor. JFrog, 2024. Backdoored pickle checkpoints discovered on Hugging Face Hub.
https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/

FORGE-06 Insecure Facility & Datacenter Management Systems

Definition

Insecure Facility & Datacenter Management Systems refers to weaknesses in the software platforms that run the physical datacenter, including the systems that control cooling, power, and physical access. These include systems like Building Management Systems, BMS, for cooling and environmental controls, Electrical Power Monitoring Systems, EPMS, for the building's electrical equipment, and Data Center Infrastructure Management, DCIM, for rack and asset tracking. Compromise of these systems can disrupt training and inference workloads, corrupt in-flight checkpoints, expose detailed maps of high-value assets, and bypass physical security, without directly compromising a workload.

Description

AI datacenters depend on facility infrastructure that is often invisible to security teams focused on servers, networks, and cloud control planes. BMS platforms control chilled water loops, computer-room air handlers, CRAH and CRAC units, and cooling systems used for high-density GPU racks. EPMS platforms monitor and sometimes control switchgear, uninterruptible power supplies, generators, and branch-circuit power distribution. DCIM platforms aggregate telemetry across the floor, track rack and asset location, and support capacity planning. Physical security systems govern who can enter the building, data hall, cages, rows, or maintenance areas.

These systems matter more for AI infrastructure than for conventional compute because GPU clusters operate closer to thermal and power limits than traditional server fleets. Modern GPU racks can draw 50 to 130 kW or more, well above the 5 to 15 kW common in earlier generations. A facility-side action that disrupts cooling or power can stop a multi-day training run, corrupt checkpoints across many nodes, or force emergency shutdown of an entire hall, without directly interacting with the compute stack.

Facility systems are often less hardened than server-side infrastructure. Legacy OT protocols such as BACnet, Modbus, and SNMPv1 are still common and may lack authentication or encryption by default. Vendor and integrator remote access can be broad or long-lived, and patch cycles are often slower than for IT systems.
The exposure varies by deployment. It is highest in legacy installations, default configurations, abandoned or unpatched products, broad vendor-access setups, and environments where facility and IT security are not jointly owned or verified.

Impact and Failure Modes

Legacy building-control protocols with no built-in security

BACnet, Modbus, and SNMPv1 are still widely used to control cooling and power. In their common legacy form they do not authenticate commands or encrypt traffic, so anyone who reaches the network can read sensors, change setpoints, or send shutdown commands. Authenticated variants exist but are not yet the norm in deployed facilities.

Always-on remote access for outside vendors

Controls integrators and equipment vendors are often given persistent remote access into facility networks, sometimes through shared accounts or broad VPN access. A compromised vendor account, support portal, or remote-access path can become a route into facility controls across one or more sites.

Internet-exposed management interfaces

Public research has identified tens of thousands of datacenter management interfaces, including DCIM platforms, cooling controllers, UPS units, and rack monitors, reachable directly from the internet. Exposure is especially dangerous when combined with default credentials, weak authentication, or missing VPN and jump-host controls.

Facility software maintained by small teams or single vendors

BMS, EPMS, DCIM, and related facility platforms may be maintained by small vendor teams, integrators, or open-source communities. When a product is abandoned, customized heavily, or no longer receives security updates, operators may be left without a clear patch path or migration plan. openDCIM, a widely deployed open-source platform, lost its lead maintainer in 2025 with no successor announced.

Prevention and Mitigation Strategies

For providers and datacenter operators

1. **Treat facility systems as a separate, segmented network:** Place BMS, EPMS, DCIM, and physical security systems on dedicated networks that are not reachable from tenant workloads, corporate IT, or the public internet. Do not assume that being inside the datacenter perimeter is enough.

2. **Govern vendor remote access:** Inventory every remote connection used by integrators, controls vendors, and equipment manufacturers. Replace always-on connections with time-bound, on-demand access through a controlled jump host. Require unique accounts per person, multi-factor authentication, and full session logging. Treat vendor access as a privileged path, not a convenience.

3. **Use authenticated and encrypted protocols where supported:** Where the equipment supports it, prefer modern variants such as BACnet/SC and Modbus Security. Replace SNMPv1 and SNMPv2c with SNMPv3 for monitoring. Where legacy protocols cannot be replaced, isolate them on dedicated network segments and restrict which devices can speak them.

4. **Restrict and monitor DCIM access:** A DCIM platform is a detailed map of high-value assets and should be protected accordingly. Apply least-privilege roles, multi-factor authentication, and audit logging for all access. Limit which users, hosts, and integrations can read or export full inventory data.

5. **Patch facility software and track lifecycle:** Maintain an inventory of BMS, EPMS, DCIM, and physical security software versions, including which products are still actively maintained by their vendors. Apply vendor patches on a defined schedule, and plan ahead for products that are abandoned or near end-of-life.

For customers evaluating a provider

Facility infrastructure sits almost entirely on the provider side and is rarely visible from inside a rented node, so this is about checking what the provider publishes and asking directly where that documentation is silent.

1. **Check the provider's security and trust documentation:** Review the provider's trust or security pages, whitepapers, and compliance scope for how facility and physical infrastructure is covered. Facility, power, and environmental controls are often in scope for these reports, so their presence, or absence, is a useful signal.

2. **Ask directly where the documentation is silent:** Published material rarely covers everything, so ask the provider how facility systems (BMS, EPMS, DCIM) are segmented from tenant and public networks, how integrator and equipment-vendor remote access is governed (time-bound, individually attributed, logged), and whether facility management interfaces are internet-exposed, given documented mass exposure of these interfaces.

Attack Scenarios

Vendor Compromise to Cooling Disruption

A controls integrator that provides remote support to several datacenter operators is breached through an unpatched public-facing maintenance portal. The integrator maintains always-on VPN connections into each customer’s facility network, so once inside the integrator’s environment, the attacker can reach the Building Management System at a GPU site without breaching the operator directly. From the BMS, the attacker lowers cooling capacity in a specific data hall by changing chilled-water setpoints and disabling several air handlers. The affected environment becomes unstable, causing workload interruptions and reduced cluster availability.

Internet-Exposed DCIM as a Map for a Targeted Attack

A DCIM platform is reachable from the public internet because it was made accessible for remote facilities staff and was never placed behind a VPN or jump host. The attacker discovers the instance through internet-wide scanning and logs in using weak or default credentials. The DCIM inventory reveals which racks belong to which tenants, where high-density GPU clusters are located, and which power feeds and cooling zones serve them. The attacker uses this information to plan a targeted disruption or physical-access attempt against a specific high-value tenant.

References

Data Centers Facing The Risk of Cyberattacks. Cyble Research Labs. Investigation that found more than 20,000 internet-exposed datacenter management interfaces, including DCIM platforms, cooling systems, UPS units, rack monitors, and automatic transfer switches, many secured only by default credentials.
https://cyble.com/blog/data-centers-facing-risk-of-cyberattacks/

NIST SP 800-82 Rev. 3, Guide to Operational Technology (OT) Security. NIST. Foundational guidance for securing industrial control and building automation systems, including segmentation, access control, and protocol-specific recommendations.
https://csrc.nist.gov/pubs/sp/800/82/r3/final

ISA/IEC 62443 Series, Security for Industrial Automation and Control Systems. ISA/IEC. International standard for OT security covering network segmentation, access control, and lifecycle management of control system components.
https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards

ASHRAE Standard 135, BACnet. ASHRAE. The base BACnet standard, used by most building management systems for HVAC and environmental control.
https://www.ashrae.org/technical-resources/bookstore/standard-135

BACnet Secure Connect (BACnet/SC). ASHRAE. Modern transport addition to BACnet that provides TLS-based authentication and encryption.
https://www.ashrae.org/file%20library/technical%20resources/bookstore/bacnet-sc-overview.pdf

Modbus Security Specification. Modbus Organization. Specification adding TLS-based authentication and encryption to the Modbus protocol.
https://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf

Target 2013 Data Breach Analysis. Senate Commerce Committee staff report. Documents how an HVAC vendor's compromised remote access was used as the initial access vector into the Target retail network.
https://www.commerce.senate.gov/services/files/24d3c229-4f2f-405d-b8db-a3a67f183883

ICS-CERT Advisories on Building Management and Power Systems. CISA. Ongoing vendor advisories covering vulnerabilities in BMS, EPMS, UPS, and DCIM products from manufacturers including Schneider Electric, Siemens, Tridium, and Vertiv.
https://www.cisa.gov/news-events/cybersecurity-advisories

openDCIM Project. Open-source DCIM platform widely deployed in academic and commercial datacenters. The project's lead maintainer announced retirement in 2025 with no successor identified at the time of writing, illustrating the maintenance fragility of facility software.
https://github.com/opendcim/openDCIM

FORGE-07 Insecure Data and Artifact Handling

Definition

Insecure Data and Artifact Handling occurs when weak controls allow attackers, insiders, compromised workloads, or privileged operators to access, copy, modify, or export sensitive AI data and derived artifacts. This includes training and fine-tuning datasets, evaluation data, embeddings, annotations, prompts, logs, checkpoints, model weights, adapters, tokenizer files, cached copies, backups, and packaged artifacts used across AI development and deployment.

Description

AI systems depend on high-value data and derived artifacts that are collected, stored, transformed, moved, cached, and reused across the AI lifecycle. This includes training and fine-tuning datasets, evaluation data, annotations, human feedback, prompts, logs, embeddings, vector indexes, checkpoints, adapters, tokenizer files, packaged model artifacts, and backups. These assets may contain sensitive customer information, proprietary business knowledge, regulated data, model behavior, or the results of significant engineering and compute investment.

The risk is broader than model theft. AI data exposure can reveal training sources, user activity, sensitive information, proprietary methods, or model behavior, while modified data or artifacts can poison downstream training, fine-tuning, evaluation, or deployment. The mechanics may resemble ordinary data exfiltration or tampering, but the impact is AI-specific because these assets can be reused, reconstructed, or silently influence future systems.

This risk arises when AI data and artifacts are stored, transferred, backed up, cached, mounted, or loaded into runtime environments without strong access controls, encryption, tenant isolation, integrity validation, lifecycle management, and auditability. In AI infrastructure, this includes datasets, object stores, model registries, experiment systems, local caches, backups, and high-performance or distributed storage systems.

If these paths are weakly isolated, over-permissioned, unaudited, or poorly cleaned between workloads or tenants, a compromised identity, compute node, tenant workload, notebook, data pipeline, storage client, registry, or deployment system can turn an initial foothold into long-term loss of confidentiality, integrity, provenance, and control.

Impact and Failure Modes

Unprotected model storage and backups

Model weights, checkpoints, or packaged model files stored in shared file systems, object stores, snapshots, or backup locations can be copied by anyone with read access. Secondary storage locations often receive weaker controls than primary model repositories.

Overbroad model registry or artifact-store access

Model registries and artifact stores can hold approved production models, deployment metadata, adapters, tokenizer files, evaluation outputs, and packaged artifacts. Weak authorization or broad service-account permissions can allow unauthorized users or workloads to download or modify valuable assets.

Multi-tenant AI storage exposure

High-performance AI storage platforms often host filesystems for multiple tenants or workloads on the same cluster. If client authentication is disabled, network reachability alone can be enough for a compromised host or malicious tenant to mount filesystems outside its own workspace, exposing other tenants' datasets, model weights, checkpoints, and training outputs, and allowing silent modification of training data or checkpoints.

Exposure to privileged infrastructure operators

When models or datasets run on third-party GPU infrastructure, provider operators or platform administrators may be able to access model artifacts, host memory, node storage, logs, or deployment paths depending on the platform trust boundary.

Detection blindness to model artifact movement

Traditional DLP and storage monitoring may not classify files such as .pt, .pth, .safetensors, .onnx, or .h5 as sensitive IP by default, allowing large transfers to blend into routine deployment, backup, or storage activity.

Storage substrates left unencrypted or unaudited

AI storage platforms generally support encryption at rest, per-filesystem audit logging, and managed TLS on the storage management plane, but these are not always enabled or correctly configured in practice. Where they are not, exfiltration can leave no forensic trail, decommissioned media may remain readable, and the management API can be intercepted or impersonated by an attacker on the storage network.

Prevention And Mitigation Strategies

For providers and datacenter operators

1. **Enforce authentication and tenant isolation at the storage layer:** Require client authentication for all filesystem mounts and storage protocols on AI storage platforms. Enforce tenant separation at the storage layer itself, not only in the scheduler or model registry. Restrict storage-network reachability only to approved hosts.

2. **Encrypt and audit the storage substrate:** Enable encryption at rest on clusters holding model artifacts and training data, and log high-impact events such as mounts, deletes, permission changes, snapshots, and administrative actions to an independent system the storage administrator cannot reach or modify. Full per-read logging is often impractical at scale, so reserve detailed access logging for the most sensitive filesystems.

3. **Clean temporary and cached copies between tenants:** Limit retention of model artifacts on training and serving nodes, clear node-local caches between tenants or jobs, and ensure snapshots and backups inherit the same access controls as primary repositories.

4. **Limit privileged infrastructure access to tenant artifacts:** Restrict operator, administrator, and service-account access to model artifacts, host memory, node storage, and deployment paths. Use short-lived credentials, scoped workload identities, and separation of duties for systems that can retrieve or deploy models.

For customers:

1. **Inventory model artifacts as high-value assets:** Track weights, checkpoints, adapters, datasets, tokenizer files, packaged files, snapshots, backups, and cached copies, and know where each exists, who can access it, and which systems can move or deploy it.

2. **Enforce least privilege on your model storage and registries:** Apply granular access controls to repositories, object stores, file systems, backup locations, and registries, with separate permissions for reading, exporting, approving, deleting, and deploying artifacts.

3. **Require approval and audit for high-impact actions:** Make downloads, exports, permission changes, model promotions, backup restores, and cross-boundary transfers generate tamper-resistant records and require approval where they could expose production or frontier models.

4. **Monitor for unusual artifact movement:** Alert on bulk downloads, unexpected copies, transfers to new destinations, and large movements of model file types such as .pt, .pth, .safetensors, .onnx, and .h5, which traditional DLP may not classify as sensitive by default.

5. **Ask how the provider protects the storage beneath your models:** When models run on third-party infrastructure, the storage layer is not fully visible from inside your workload. Ask whether the provider requires client authentication, separates tenants at the storage layer, encrypts at rest, and audits storage access. Treat a provider that cannot answer as carrying residual risk.

Attack Scenarios

Exfiltration via Unsecured Backup

An attacker discovers model checkpoints stored in backup, snapshot, or mirror locations that do not enforce the same access controls as the primary model repository. Because many security programs focus on source code and traditional sensitive data, model artifacts in secondary storage may be poorly inventoried or insufficiently monitored. Large outbound transfers of model files can blend into normal backup or replication activity and may not be flagged unless model artifact movement is explicitly monitored.

Cross-Tenant Model Theft Through Unauthenticated AI Storage

A customer of a neocloud GPU provider receives access to a compute node connected to a shared high-performance AI storage cluster, such as Weka, VAST Data, DDN, Lustre, BeeGFS, CephFS, or a similar platform. The cluster hosts filesystems for multiple tenants, but filesystem client authentication is disabled or left permissive. From their own node, the customer can mount another tenant's filesystem and copy model weights, checkpoints, datasets, and training outputs. Because access logging is not enabled on that filesystem, the provider has limited record of which files were accessed or copied. If write access is also permissive, the same path can be used to tamper with training data or checkpoints. These platforms support client authentication, storage-layer tenant isolation, and audit logging, so the exposure concentrates in deployments where those controls are off or unconfigured.

Compromised Deployment Service Account

An attacker compromises a service account used by a model deployment pipeline. The account is allowed to pull approved model files from a registry and copy them to serving infrastructure. The attacker uses the same trusted path to download model weights outside the approved environment. Because the activity resembles normal deployment traffic, defenders may not notice the theft unless model artifact movement is explicitly monitored.

Model Cache Left on a GPU Serving Node

A serving node keeps local copies of model files to reduce startup time. After a workload is removed or reassigned, the cached files remain on disk or in a location readable by another process. A compromised workload or later tenant accesses the leftover cache and copies model artifacts that were never meant to be available outside the original serving environment.

References

MITRE ATLAS. MITRE. Adversarial threat landscape for AI systems.
https://atlas.mitre.org/

NIST AI Risk Management Framework. NIST. Framework for managing AI-related risks.
https://www.nist.gov/itl/ai-risk-management-framework

Stealing Machine Learning Models via Prediction APIs. Tramèr et al., USENIX Security 2016. Demonstrates model extraction through query access to prediction APIs.
https://arxiv.org/pdf/1609.02943

Extracting Training Data from Large Language Models. Carlini et al., USENIX Security 2021. Shows that LLMs memorize and can leak training data.
https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting

Hermes Attack: Steal DNN Models with Lossless Inference Accuracy. Zhu et al., USENIX Security 2021. GPU side-channel attack enabling model extraction.
https://www.usenix.org/conference/usenixsecurity21/presentation/zhu

NVIDIA Trusted Computing Solutions. NVIDIA. Confidential computing for GPU workloads.
https://docs.nvidia.com/confidential-computing/index.html

FORGE-08 Certification Gaps & Provider Transparency Failures

Definition

Certification Gaps & Provider Transparency Failures occur when customers rely on provider certifications, attestations, or security claims that do not clearly apply to the actual AI infrastructure running their workloads. This risk is amplified when providers do not disclose audited scope, upstream operators, reselling arrangements, subservice organizations, regional differences, or which controls apply to each environment.

Description

Many GPU cloud providers present security assurance through certifications such as SOC 2 or ISO 27001, but those attestations do not automatically prove that the full AI infrastructure stack has been assessed. In practice, audit scope may cover the provider's customer portal, API, and internal software processes while excluding the physical GPU hosts, backend fabric, BMC environment, firmware lifecycle, node reassignment procedures, or upstream infrastructure operators.

This gap becomes harder to detect when provider transparency is limited. Some providers resell or aggregate capacity from other operators, lease infrastructure from third parties, or rely on subservice organizations that are not obvious to the customer. Regional differences can also matter if controls, operators, or audited environments vary by location. HPC security guidance such as NIST SP 800-223 and NIST SP 800-234 can help customers ask better questions about specialized hardware, high-speed networks, architecture, and security controls. But these documents are guidance, not provider-specific assurance.

They do not prove that a given provider implemented the controls or that the audited scope covers the infrastructure actually running the workload. The result is a false sense of assurance. Customers may believe a certified provider has independently validated the security of the full AI environment, when the attested scope is partial or unclear. This challenge is especially important in fast-growing GPU cloud environments where security documentation, control standardization, and infrastructure transparency may lag behind deployment.

Impact and Failure Modes

Certification scope excludes AI infrastructure

A provider holds SOC 2 or ISO 27001, but the assessed scope does not include GPU hosts, network fabric, BMC controls, firmware integrity, or node lifecycle operations.

No penetration testing of the infrastructure

Some providers, including both emerging neoclouds and established vendors, do not conduct regular penetration testing of their AI infrastructure environments, meaning fundamental security weaknesses may go undiscovered until exploited.

Undisclosed upstream operators

A customer contracts with one provider, but the workload runs on infrastructure operated by another entity whose controls, patching cadence, or incident response processes are not clearly disclosed.

Inconsistent assurance across aggregated capacity

A provider combines capacity from multiple operators with different security controls, but presents a single trust posture to customers.

Regional or subservice gaps

Security controls and attestations differ by geography, vendor, or subservice organization, but customers are not clearly told which environments fall outside the assurance boundary.

False confidence from narrow attestations

Customers rely on self-attestation, SOC 2 Type I, or marketing claims as if they prove ongoing operational effectiveness across the full AI stack.

Prevention and Mitigation Strategies

For providers and datacenter operators

1. **Make attestation scope explicit:** State clearly which parts of the stack a certification actually covers, so customers cannot mistake a portal-and-API audit for full-stack assurance.

2. **Disclose infrastructure sourcing:** Disclose whether capacity is owned, leased, brokered, or resold, and identify the upstream operators or subservice organizations involved, including regional differences in controls.

3. **Test the infrastructure, not just the software:** Conduct regular penetration testing and security review of the GPU infrastructure environment, not only the customer portal and API.

For customers evaluating a provider

1. **Review attestation scope carefully:** Verify whether certifications and audit reports include GPU infrastructure, backend fabric, management-plane controls, firmware lifecycle, and tenant reassignment procedures, not just the API or software layer.

2. **Require transparency about infrastructure sourcing:** Contracts and security documentation should disclose whether capacity is owned, leased, brokered, or resold, and identify the upstream operators or subservice organizations involved.

3. **Prefer stronger independent assurance where needed:** For production or regulated workloads, require meaningful third-party assurance such as SOC 2 Type II or ISO 27001 with operationally relevant scope rather than relying only on self-attestation or point-in-time reviews.

4. **Verify the AI-specific controls, not just the certificate:** Use a targeted questionnaire or control review of the AI infrastructure itself, not just the general certification. The CSA AI-CAIQ is a useful starting point.

5. **Verify the machine is operated by the provider you expect:** Confirm that your assigned infrastructure traces back to the provider you contracted with, not an undisclosed upstream operator. Public ownership records for your node's external IP, reverse DNS, and the TLS certificates on management or API endpoints can all indicate who actually operates it. A mismatch is not proof of a problem, since infrastructure is often registered to a parent or upstream operator, but it is worth reconciling against what the provider disclosed.

Attack Scenarios

Certification Scope Mismatch

An enterprise selects a GPU cloud provider based on SOC 2 Type II and marketing claims of secure AI infrastructure. After a security incident, the customer discovers that the audited scope covered only the provider’s orchestration platform and customer portal. The physical GPU systems, BMC environment, backend fabric, and storage layer were operated by a third party outside the assessed boundary, leaving the customer without evidence that the actual workload environment met the expected controls.

Hidden Upstream Provider

A provider aggregates capacity from multiple upstream operators but does not clearly disclose that arrangement. A customer assumes the advertised security posture applies uniformly across all regions and clusters. After an incident, the investigation shows that the affected workload ran on infrastructure managed by a different operator with weaker lifecycle controls and different patching practices, creating a gap between the promised assurance and the actual environment.

References

NIST SP 800-234 High-Performance Computing (HPC) Security Overlay

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-234.pdf

NIST SP 800-223 High-Performance Computing Security: Architecture, Threat Analysis, and Security Posture
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-223.pdf

AICPA SOC 2 / Trust Services Criteria. AICPA. Audit standard for service organization controls.
https://www.aicpa-cima.com/topic/audit-assurance/audit-and-assurance-greater-than-soc-2

ISO/IEC 27001. ISO. Information security management system standard.
https://www.iso.org/standard/27001

Cloud Controls Matrix (CCM). Cloud Security Alliance. Cloud-specific security control framework.
https://cloudsecurityalliance.org/research/cloud-controls-matrix

CAIQ / STAR Level 1 Security Questionnaire. Cloud Security Alliance. Standardized cloud security self-assessment.
https://cloudsecurityalliance.org/artifacts/star-level-1-security-questionnaire-caiq-v4-1

STAR Registry. Cloud Security Alliance. Public registry of cloud provider security postures.
https://cloudsecurityalliance.org/star/registry

NIST SP 800-161 Rev. 1, Cybersecurity Supply Chain Risk Management. NIST. Supply chain risk management practices for systems and organizations.
https://csrc.nist.gov/pubs/sp/800/161/r1/upd1/final

Shared Assessments, SIG Questionnaire. Shared Assessments. Standardized third-party risk assessment.
https://sharedassessments.org/sig/

AI Consensus Assessments Initiative Questionnaire (AI-CAIQ). Cloud Security Alliance. AI-specific cloud security assessment.
https://cloudsecurityalliance.org/artifacts/ai-consensus-assessments-initiative-questionnaire-ai-caiq

FORGE-09 Insecure Operational Infrastructure Services

Definition

Insecure Operational Infrastructure Services refers to compromise of the privileged platforms and automation used to operate AI infrastructure, including job schedulers, cluster control planes, server-preparation workflows, configuration automation, and internal administrative interfaces. These services decide what runs, where it runs, and how GPU systems are prepared and maintained. A single compromise can give attackers broad control over training and inference environments.

Description

AI infrastructure depends on an operational layer that turns GPU servers into usable training and inference capacity. Schedulers and cluster control planes such as Slurm, Kubernetes, and Ray decide where jobs run and what resources they receive. Server-preparation workflows install or configure the operating environment, GPU drivers, container runtime, monitoring agents, and baseline security settings before workloads are placed on nodes. Configuration automation, such as Ansible or internal fleet-management tooling, applies system settings, runtime configuration, and security controls across large fleets. Internal administrative tools expose these actions to operators through APIs, dashboards, and support workflows.

This risk arises when schedulers, provisioning systems, admin tools, automation pipelines, or cluster-management services are exposed, weakly authenticated, over-permissioned, or insufficiently segmented from tenant and enterprise environments. A compromised administrator account, SaaS integration, automation token, CI/CD system, or service credential can turn these trusted operational paths into attacker-controlled paths inside the AI infrastructure environment.

A compromise of these privileged services is severe because they hold broad authority over high-value GPU nodes and workloads. By controlling schedulers or configuration systems, an attacker can manipulate workloads, push malicious changes, weaken isolation, disable monitoring, alter node state, or compromise the integrity of rebuilt systems across the infrastructure.

Impact and Failure Modes

Compromised scheduler or orchestration controller

An attacker who controls Slurm, Kubernetes, Ray, or similar orchestration services may be able to submit jobs, modify workloads, change placement decisions, stop training runs, or access operational metadata across the cluster.

Exposed or weakly controlled job-submission services

Job-submission APIs such as Ray Jobs can become direct execution paths into the cluster. If exposed without appropriate controls, attackers can use them to run unauthorized workloads on GPU infrastructure.

Configuration automation blast radius

Configuration automation such as Ansible or internal fleet-management tooling often has broad administrative reach. Compromise of one automation path can push malicious settings, weaken isolation, disable monitoring, alter runtime configuration, or deploy persistence across many GPU nodes.

Privileged internal admin interface abuse

Internal dashboards and platform APIs that wrap scheduler, server-preparation, or lifecycle functions can become direct control surfaces. Weak authentication, broad operator roles, or insufficient approval for high-impact actions can turn a single admin interface compromise into cluster-wide operational control.

Node-preparation pipeline compromise

Server images, bootstrap scripts, driver installers, startup templates, and node preparation workflows define the trusted baseline for GPU hosts. If compromised, they can introduce backdoors, insecure configuration, vulnerable drivers, or disabled security controls into every node built or rebuilt from that pipeline.

Service credential and automation token exposure

Tokens used by CI/CD systems, SaaS integrations, deployment tools, schedulers, or automation services may carry broad infrastructure permissions. If leaked or over-scoped, they can allow attackers to submit jobs, modify configuration, pull artifacts, or access operational systems without compromising an interactive user account.

Prevention And Mitigation Strategies

For providers and datacenter operators

1. **Block tenant access to operational services:** Tenant workloads, notebooks, containers, and ordinary job networks should not be able to reach schedulers, cluster control-plane APIs, configuration automation, node-preparation services, or privileged admin interfaces.

2. **Do not expose operational services broadly:** Operational services should not be reachable from the public internet or broad corporate networks. Restrict access to dedicated management networks, approved VPNs, bastion hosts, or controlled administrative workstations.

3. **Enforce strong authorization for operational actions:** Apply least privilege, RBAC, MFA where supported, and separation of duties for actions that submit jobs, alter cluster state, promote images, push configuration, or return nodes to service.

4. **Require approval and tamper-resistant audit records for fleet-wide actions:** Configuration pushes, server image changes, scheduler policy changes, and broad administrative actions should require explicit approval and generate audit records suitable for incident investigation.

5. **Protect node build and bootstrap files:** Restrict who can modify the operating-system images, startup scripts, and templates used to prepare GPU nodes, and require review before changes are promoted.

6. **Monitor operational service behavior:** Alert on unusual job submissions, node build changes, configuration pushes, scheduler policy changes, and administrative access.

For customers

1. **Do not expose your own job-submission and orchestration endpoints:** If you run Ray, a Kubernetes dashboard, a Jupyter or notebook server, or similar interfaces on rented GPUs, keep them off the public internet and behind authentication. Exposed orchestration endpoints (the ShadowRay pattern) are one of the most actively exploited paths into AI workloads, and on rented infrastructure they are usually yours to secure, not the provider's.

2. **Authenticate and scope your own automation:** Require authentication on any job-submission API you operate, and scope the credentials and tokens used by your CI/CD, automation, and SaaS integrations so a single leaked token cannot reach your whole environment.

3. **Check that nothing operational is exposed to the internet:** From outside, confirm that your rented machines' external IPs do not expose schedulers, dashboards, job-submission APIs, admin interfaces, or metrics endpoints without authentication. The ShadowRay pattern is exactly this kind of exposure, found from the internet.

Attack Scenarios

ShadowRay: Ray Cluster Abuse

Ray is an open-source framework for running distributed AI and Python workloads, and its dashboard and Jobs API are used to manage and submit work to Ray clusters. An attacker discovers an exposed Ray dashboard or job-submission interface used for AI workloads. Because the interface is reachable without appropriate access controls, the attacker submits unauthorized jobs through the same operational path used to run distributed workloads. Those jobs can consume GPU capacity, run attacker-controlled code, access environment variables or operational metadata available to the workload, and create a foothold for follow-on activity. Oligo documented this pattern in its ShadowRay research on exposed Ray environments used for AI workloads

Configuration Automation Compromise

An attacker compromises Ansible or an internal fleet automation tool used to manage GPU nodes. They push a change that disables security controls, adds an unauthorized key, or alters container runtime configuration. Because these tools are designed to apply changes across many systems, a single compromise can affect many nodes before defenders notice.

Scheduler and Automation Pivot from a Low-Privilege Workload

An attacker compromises a low-privilege container used for preprocessing or job setup. Because the environment reuses service-account credentials and does not strongly separate workload, scheduler, and automation paths, the attacker pivots into job-control systems or supporting operational services. From there, they can influence workload placement, access model storage, alter jobs, or reach administrative paths that affect the broader AI environment. A limited workload compromise becomes a wider operational compromise because trusted infrastructure services are over-permissioned or weakly segmented.

References

NIST SP 800-223, High-Performance Computing Security: Architecture, Threat Analysis, and Security Posture. NIST. HPC security architecture, threat model, and posture guidance relevant to AI infrastructure operations.
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-223.pdf

NIST SP 800-234, High-Performance Computing (HPC) Security Overlay. NIST. Security control overlay for HPC environments, including systems used for AI and machine learning workloads.
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-234.pdf

CIS Kubernetes Benchmark. Center for Internet Security. Kubernetes hardening benchmark.
https://www.cisecurity.org/benchmark/kubernetes

Ray Security Documentation. Ray. Security guidance for Ray clusters and interfaces.
https://docs.ray.io/en/latest/ray-security/index.html

Ray Jobs API. Ray. API for submitting and managing jobs on remote Ray clusters.
https://docs.ray.io/en/latest/cluster/running-applications/job-submission/api.html

ShadowRay: AI Workloads Actively Exploited in the Wild. Oligo Security, 2024. Reporting on exposed Ray environments abused against AI workloads.
https://www.oligo.security/blog/shadowray-attack-ai-workloads-actively-exploited-in-the-wild

Slurm Authentication. SchedMD. Authentication guidance for Slurm workload manager deployments.
https://slurm.schedmd.com/authentication.html

FORGE-10 Vendor Embargo Gaps & Patch Velocity Failures

Definition

Vendor Embargo Gaps & Patch Velocity Failures refer to the risk that data centers and AI infrastructure remain exposed to known vulnerabilities because some providers receive no advance notice before public disclosure, while others patch too slowly or inconsistently after fixes become available. In GPU environments, this creates exploitable windows between disclosure and remediation, amplified by disruptive update procedures, firmware complexity, and version drift across large fleets.

Description

The speed at which GPU cloud providers deploy security fixes varies widely. Some providers may receive earlier notice or mitigation guidance through coordinated disclosure, OEM, or vendor-partner channels, while others learn of critical issues only when public bulletins or CVEs are released. Providers without advance coordination can therefore face a zero-notice exposure window in which a vulnerability is public before their fleet is patched.

Recent NVIDIA Container Toolkit vulnerabilities illustrate the problem. Publicly disclosed issues such as CVE-2024-0132 and CVE-2025-23266 showed that vulnerabilities in core GPU container infrastructure can enable privilege escalation or container escape on systems running affected versions. Once technical details or public exploit guidance are available, the barrier to exploitation drops sharply for any provider still running vulnerable toolkit versions.

Even when fixes exist, GPU environments create strong incentives to delay deployment. Driver and toolkit updates can require node drains and job disruption. Firmware updates are operationally harder and are often treated as risky maintenance. Over time, this leads to version fragmentation across large fleets, where some nodes are updated quickly and others continue running older, vulnerable software. Without clear remediation targets, customer visibility, and version enforcement, exposed nodes can persist long after a fix is available.

See also: FORGE-05 AI Infrastructure Supply Chain Compromise for dependency-level risks that intersect with patch velocity.

Impact / Examples

Zero-notice exposure after public disclosure

Providers without advance notice or pre-coordinated mitigation guidance may be exposed from the moment a vulnerability becomes public.

Public exploit guidance lowers the cost of attack

Once CVEs, advisories, technical writeups, or proof-of-concept techniques are published, attackers can target unpatched GPU environments with less effort.

Workload disruption encourages patch deferral

Providers may delay driver, toolkit, or runtime updates because patching interrupts active training and inference jobs.

Firmware and low-level component updates are avoided or postponed:

Operational complexity, maintenance windows, and fear of service impact can leave BMC, BIOS/UEFI, NIC, GPU, storage, or platform firmware unpatched for long periods.

Fleet version drift creates a persistent attack surface

Heterogeneous versions across large clusters can leave a subset of nodes vulnerable even after patches are available.

Prevention and Mitigation Strategies

For providers and datacenter operators

1. **Establish internal patch SLAs by severity:** Define concrete remediation targets for critical, high, and medium vulnerabilities, and treat GPU container runtimes, drivers, firmware, and orchestration components as security-critical assets.

2. **Participate in coordinated vendor security channels where possible:** Subscribe to vendor bulletins and use available customer, OEM, or partner security-notification paths to reduce zero-notice exposure.

3. **Track security-relevant software and firmware versions continuously:** Monitor versions of components such as nvidia-container-toolkit, GPU Operator, GPU drivers, device firmware, and comparable AMD stack components across all nodes as key security indicators.

4. **Pre-stage rolling deployment pipelines:** Use canary deployments, rolling node drains, and maintenance automation so critical patches can be applied quickly without improvised operations.

5. **Integrate firmware updates into lifecycle management:** Treat firmware patching as a routine part of node intake, servicing, and retirement rather than exceptional maintenance.

6. **Provide customer-visible version and patch status where appropriate:** Expose current runtime, driver, and firmware versions, or at least patch status, so customers can assess residual risk.

7. **Include remediation expectations in provider agreements:** Where contractually possible, define expected timelines for patching critical vulnerabilities and communicating material exposure.

For customers and teams running their own GPU stack

Part of this risk is the provider's to manage, but on most offerings you run your own container toolkit, drivers, and images, so that part is yours to patch.

1. **Patch the GPU stack you control promptly:** Track and update the components you run yourself, such as the NVIDIA Container Toolkit, GPU Operator, drivers, and your container base images, against advisories like the NVIDIA Container Toolkit security bulletins.

2. **Subscribe to vendor security notifications:** Follow NVIDIA, AMD, and relevant OEM advisories directly so you learn about critical GPU-stack vulnerabilities when they are disclosed rather than after an incident.

3. **Ask the provider for version and patch visibility:** Ask whether the provider exposes current runtime, driver, and firmware versions, or at least patch status, for the layers it manages, and what its target timelines are for patching critical vulnerabilities.

4. **Put remediation expectations in the contract:** Where you can, define expected timelines for patching critical vulnerabilities and for being notified of material exposure on the provider-managed layers.

5. **Run vulnerability scans:** Scan all systems regularly for known vulnerabilities and send the report to owners and to providers. Establish a policy for the customer organization that critical vulnerabilities need to be fixed in 7 days, high ones in 14 days, and systems running with critical vulnerabilities after the patch window will be taken off the network.

Attack Scenarios

Mass Exploitation During the Disclosure Window

A critical container-runtime vulnerability is publicly disclosed with enough technical detail to guide exploitation. Providers with affected GPU container hosts but no advance warning remain unpatched when the bulletin becomes public. An attacker rapidly tests reachable or customer-accessible environments across multiple providers and compromises hosts still running vulnerable toolkit versions.

Exploiting Fleet Version Drift

An attacker targets a large GPU cluster knowing that version inconsistency is common. They probe for nodes running older runtime or toolkit versions and focus exploitation on that vulnerable subset. Because the provider lacks strong version enforcement, a minority of nodes remains exploitable long after patches were released.

Exploiting Fleet Version Drift

An attacker targets a large GPU cluster knowing that version inconsistency is common. They probe for nodes running older runtime or toolkit versions and focus exploitation on that smaller vulnerable subset. Because the provider lacks strong version enforcement, a minority of nodes remains exploitable long after patches were released.

References

CVE-2024-0132, NVIDIA Container Toolkit container escape (CVSS 9.0). https://nvd.nist.gov/vuln/detail/CVE-2024-0132

NVIDIA Container Toolkit Security Advisories. NVIDIA. Toolkit vulnerability disclosures and fixes. https://github.com/NVIDIA/nvidia-container-toolkit/security/advisories

CVE-2025-23266, NVIDIA Container Toolkit privilege escalation (NVIDIAScape). https://nvd.nist.gov/vuln/detail/CVE-2025-23266

NVIDIA PSIRT Policies. NVIDIA. Coordinated disclosure and embargo practices. https://www.nvidia.com/en-us/security/psirt-policies/

NVIDIA Product Security Notifications. NVIDIA. Security bulletin and advisory feed. https://www.nvidia.com/en-us/security/

AMD Product Security. AMD. Security advisories for AMD GPU and infrastructure components. https://www.amd.com/en/resources/product-security.html

NIST SP 800-40 Rev. 4, Guide to Enterprise Patch Management Planning. NIST. Patch management lifecycle guidance. https://csrc.nist.gov/pubs/sp/800/40/r4/final

NVIDIA AI Vulnerability Deep Dive, CVE-2024-0132. Wiz, 2024. Technical analysis of NVIDIA Container Toolkit container escape. https://www.wiz.io/blog/nvidia-ai-vulnerability-deep-dive-cve-2024-0132

NVIDIAScape, CVE-2025-23266. Wiz, 2025. Technical analysis of toolkit privilege escalation vulnerability.https://www.wiz.io/blog/nvidia-ai-vulnerability-cve-2025-23266-nvidiascape

Thanks For Reading

