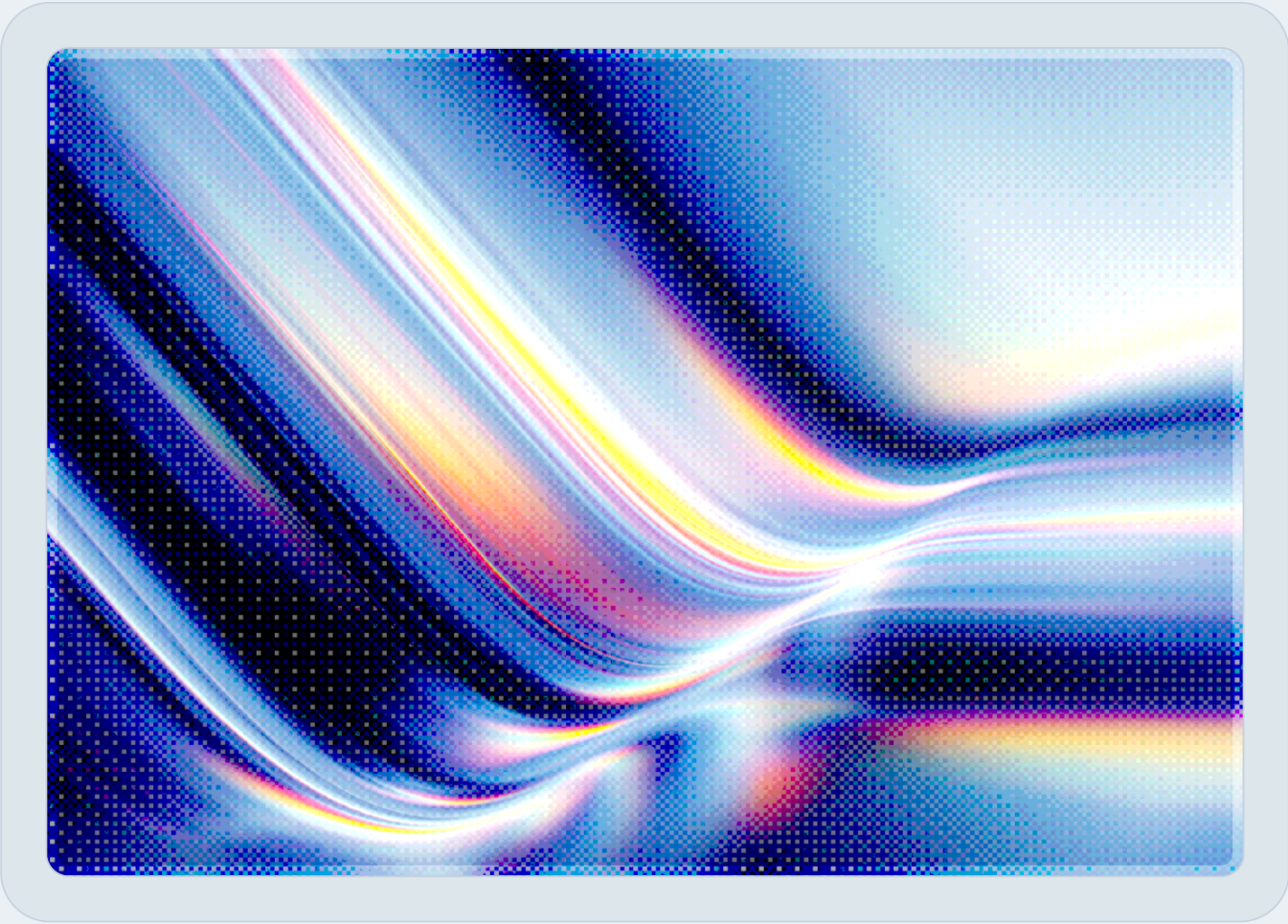
CSAI Foundation | Cloud Security Alliance

ChromaToast: Pre-Auth RCE in ChromaDB and AI Infrastructure Exposure

2026-05-21

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- CVE-2026-45829, nicknamed "ChromaToast," is a pre-authentication remote code execution vulnerability in the ChromaDB Python FastAPI server affecting every release from 1.0.0 onward. The National Vulnerability Database assigned a CVSS v4.0 base score of 10.0 and a CWE-94 (Code Injection) classification when it published the record on May 18, 2026 [1, 2].
- The flaw arises from execution ordering: the server instantiates a client-supplied embedding function — including downloading and running an attacker-controlled HuggingFace model with `trust_remote_code=True` — before the authentication check fires. By the time the server returns a 403, arbitrary Python has already executed inside the Chroma process [1, 3].
- HiddenLayer estimates that approximately 73% of internet-exposed ChromaDB instances run a vulnerable version, against a project that records roughly 13 million monthly downloads on PyPI and is in production use at organizations including Mintlify, Factory AI, and Weights & Biases [1, 4].
- A separate UpGuard survey of 1,170 internet-accessible Chroma databases found that roughly one third (around 406 instances) were exposing collection data — including customer support chat logs and personally identifiable information drawn from retrieval-augmented generation pipelines [5].
- The Chroma maintainers had not acknowledged the report or shipped a confirmed fix at the time of public disclosure. HiddenLayer attempted contact via email, GitHub, IT-ISAC, and social media between February 17, 2026 and April 16, 2026 without response, and an independent researcher had filed a similar report in November 2025 [1, 6].
- Recommended interim defenses are architectural rather than version-based: deploy the Rust server binary, terminate authentication at a reverse proxy or service-mesh sidecar so the Python process never receives anonymous requests, and restrict egress to HuggingFace from inference hosts [1, 3].

Background

ChromaDB is one of the most widely deployed open-source vector databases supporting retrieval-augmented generation (RAG) and agentic AI applications. The project's Python client and FastAPI server together account for roughly 13 million monthly PyPI downloads, and the server has become a common storage tier for embeddings in commercial and self-hosted AI stacks alike [1, 4]. Because the database sits between an enterprise's source documents and the language model consuming them, a compromised Chroma instance grants attackers visibility into the proprietary content that organizations have spent effort centralizing for retrieval — and, where stored in the process environment, the cloud credentials used by embedding providers and persistence backends.

The vulnerability category exposed by CVE-2026-45829 is not new. CSA's joint "AI Vulnerability Storm" briefing in April 2026 documented a recurring pattern in which production AI components inherit defaults and architectural assumptions from research codebases — open ports, optional authentication, dynamic loading of remote artifacts — that are unsuitable for adversarial environments [7]. ChromaToast extends that pattern from inference servers and proxies into the retrieval tier. It also illustrates a structural problem with how AI infrastructure handles untrusted configuration: parameters like `trust_remote_code` were designed for trusted research notebooks, not for HTTP endpoints reachable from the public internet.

The disclosure dynamics also matter for defenders. HiddenLayer published its analysis only after multiple unanswered contact attempts to the Chroma project, and version 1.5.9 — released roughly two weeks before public disclosure — has not been confirmed by either the maintainers or independent researchers as containing the necessary fixes [4, 6]. Enterprises evaluating their exposure therefore cannot rely on a version-bump remediation alone; they must apply compensating controls until the upstream behavior is verifiably changed.

Security Analysis

The Vulnerability Mechanics

ChromaDB's Python FastAPI server exposes a REST API for tenant, database, and collection management. The relevant endpoint for this vulnerability is `POST /api/v2/tenants/{tenant}/databases/{database}/collections`, the call used to

create a new collection. The endpoint accepts a JSON body that includes a `configuration_json` object describing the embedding function the collection should use, and the server constructs that embedding function as part of request handling [1, 3].

The defect is one of ordering. When the request arrives, the FastAPI handler calls a configuration-loading routine that instantiates the embedding function via the `sentence_transformers` integration. The configuration is permitted to specify a model name and a set of keyword arguments, and the loader passes those arguments through to the underlying `transformers` library. When `trust_remote_code=True` is supplied alongside a model identifier under the attacker's control, the `transformers` library will fetch the model's repository from HuggingFace, dynamically import its `auto_map` Python module, and execute that module within the server process. Module-level code therefore runs with the privileges of the Chroma service before the request ever reaches the authentication middleware. The authentication check executes only after the embedding function has been constructed, and even when it correctly returns 403, the payload has already executed [1, 3].

Two independent design failures compound to produce this outcome. First, the server accepts client-supplied model identifiers and dangerous keyword arguments without restriction — `trust_remote_code` passes through type validation because booleans are part of the whitelisted JSON primitives. Second, authentication is placed after configuration loading rather than before, so any control point that could reject the request is bypassed by the side effects of building the configuration object. Either flaw on its own would be a serious bug; together, they make every Python FastAPI deployment with a network-reachable port exploitable by anyone able to send an HTTP request [1].

What an Attacker Gains

Successful exploitation yields arbitrary Python execution within the Chroma process. The practical consequences track the resources the process can already reach, which in typical deployments include the vector store itself, the embedding-provider credentials Chroma uses to compute embeddings on ingest, and the credentials used to persist collections to S3, GCS, Azure Blob, or local disk [3]. For many RAG deployments, the vector store contains highly sensitive operational text — internal documentation, customer support transcripts, source code, contracts — because the value of RAG depends on indexing the content the model needs to ground its answers on.

UpGuard's earlier survey of exposed Chroma instances established that this is not a theoretical concern: among 1,170 internet-accessible Chroma databases, approximately one third exposed collection data, including customer support chat logs from e-commerce providers and personally identifiable information drawn from Canva Creator records. Roughly 60% of the exposed instances held multiple collections beyond the default, and 32% held five or more, suggesting that many of the affected systems

were in active or recently active use, not merely abandoned prototypes [5]. CVE-2026-45829 elevates the consequence of that pre-existing exposure problem from data theft to code execution: an attacker no longer needs the database to be misconfigured for read access; they need only its TCP port to be reachable.

The Hacker News separately reported on a broader scan of approximately one million exposed AI services, finding that vector databases, inference gateways, and orchestration APIs are routinely deployed to the public internet without authentication. ChromaToast sits inside that wider exposure problem, and the volume of vulnerable services means an attacker performing internet-wide scanning will find targets faster than defenders can find their own instances [8].

Disclosure and Maintainer Response

The disclosure timeline warrants scrutiny as part of the defensive picture. An independent researcher initially filed a related report against ChromaDB in November 2025; that report did not result in a public patch. HiddenLayer's Esteban Tonglet reported the specific CVE-2026-45829 condition to the Chroma project on February 17, 2026, followed up on February 24, contacted the project through IT-ISAC on March 5, and made a final attempt across all available channels on April 16 [1, 6]. NVD published the CVE record on May 18, 2026 with a CVSS v4.0 base score of 10.0 and the CWE-94 classification, and HiddenLayer's full technical write-up followed the same week [2]. The Chroma project released version 1.5.9 approximately two weeks before disclosure, but neither HiddenLayer nor independent researchers have confirmed that 1.5.9 closes the authentication-ordering defect; mainstream coverage explicitly described the patch status as "unclear" [4].

For defenders, two operational implications follow. First, this disclosure pattern — a popular AI infrastructure project receiving credible vulnerability reports without acknowledging them — is the same pattern that has appeared with other AI projects in 2026 and should be treated as a recurring failure mode rather than an isolated incident. Second, the public disclosure preceded an authoritative fix, so organizations cannot wait for an upstream patch and must instead reach the same protection through reverse proxies, network segmentation, and egress control.

Why This Pattern Recurs in AI Infrastructure

The conditions that produced ChromaToast are not specific to Chroma. AI infrastructure tooling has developed against a small set of recurring assumptions — that operators are trusted developers running in private networks, that loading remote models is a normal runtime operation, that authentication is something to be added later — that all break in production deployments. Ollama [13], Meta Llama Stack [14], vLLM [15], NVIDIA TensorRT-LLM [16], and the Model Context Protocol [17] have each produced

disclosed RCE conditions in the past 18 months that share a structural pattern: privileged operations triggered before, or independently of, authentication. CSA's April 2026 "AI Vulnerability Storm" briefing characterized the operating environment in which these disclosures occur — AI-accelerated vulnerability discovery, compressed exploitation windows, and a maintainer ecosystem still developing the response practices common in more mature software security cultures [7]. CVE-2026-45829 belongs to the same family: the relevant code path treated remote model loading as a benign configuration step and placed it on the wrong side of the trust boundary.

The economic context amplifies the risk. Sysdig's research on LLMjacking documents an active criminal economy paying for any access to AI service accounts, and the dollar value of stolen LLM access — sometimes exceeding $100,000 per day in incurred victim costs — gives well-resourced threat actors the incentive to invest in scanners purpose-built for AI infrastructure rather than generic web reconnaissance [9]. A pre-authentication RCE in one of the most widely deployed open-source vector databases is exactly the class of finding such scanners are looking for, and, given documented LLMjacking economics, the conversion path from discovery to monetization is likely short.

Indicators and Detection Opportunities

Defenders attempting to identify exploitation attempts against CVE-2026-45829 can focus on a small set of artifacts on the Chroma host. HuggingFace model downloads triggered by API requests will populate `~/.cache/huggingface/modules/transformers_modules/` with directories whose names reflect the model identifier supplied by the requester; unexpected entries here — particularly those originating from unfamiliar HuggingFace organizations or aligned in time with collection-creation activity — are direct evidence of exploitation [3]. Outbound HTTPS to `huggingface.co` from a Chroma host is a reliable network indicator of exploitation in progress; egress-based detection is harder for an attacker to suppress than content inspection of POST bodies, since suppressing the HuggingFace fetch would break the exploit itself. Process-level telemetry showing that the Chroma worker has spawned a child shell or established outbound connections to addresses other than its persistence backend is a tertiary signal that warrants immediate investigation.

API access logs that show a 403 response on a `POST` to the collections endpoint, immediately followed by anomalous process behavior, are particularly diagnostic: that combination is essentially the exploit signature, because the vulnerability returns 403 by design while the malicious code is already running.

Recommendations

Immediate Actions

Enterprises operating ChromaDB should treat any Python FastAPI deployment reachable from a network interface as compromised-pending-investigation until the authentication-ordering behavior is verified or mitigated. The first defensive priority is to remove direct network exposure of the Python server. Internet-facing instances should be taken offline or placed behind a reverse proxy that terminates authentication before the request reaches Chroma; nginx, Envoy, or a service-mesh sidecar can perform this role with relatively little configuration. The objective is that the Chroma process literally never receives a request from an unauthenticated client, because the vulnerability is exploitable on the first byte of that request.

Where operationally feasible, switch deployments to the Rust server binary (`chroma run`). The Rust implementation does not share the Python FastAPI's embedding-function instantiation path and is not vulnerable to the disclosed CVE under current analysis [1, 3]. Organizations that cannot make this switch immediately should at minimum restrict outbound network access from the Chroma host so that any attempted model download from HuggingFace fails. Blocking egress to `huggingface.co` and its subdomains from Chroma hosts neutralizes the current public exploit path even if the input validation flaw remains.

Forensic triage on any Chroma host that was internet-reachable should include enumeration of `~/.cache/huggingface/modules/transformers_modules/` directory creation events, examination of Chroma worker process trees for unexpected children, and collection of outbound network logs for connections inconsistent with documented embedding-provider endpoints. Where any indicator is present, treat the host as compromised: rotate every credential the process had access to, including embedding-provider API keys, storage backend credentials, and any cloud IAM credentials in the process environment.

Short-Term Mitigations

Authentication architecture for vector database tiers should be restructured so that no security-relevant action occurs before the request is authenticated and authorized. In the Chroma case, this means treating the Python FastAPI server as a backend that must sit behind an authenticating proxy; in general, it means assessing every component in the AI stack for code paths that load external resources during request

handling and verifying that those paths are gated by authentication. The CVE-2026-45829 root cause — that configuration loading was treated as cheap parsing rather than as a security boundary — is a pattern that exists in other AI-adjacent codebases that have not yet been audited.

Input validation should reject dangerous keyword arguments at the API layer. Even where upstream maintainers do not strip `trust_remote_code`, `use_auth_token`, and `code_revision` from incoming requests, a proxy or WAF rule can do so. This is a stop-gap rather than a solution — the proper fix is to remove the dynamic-code-loading capability from the request path entirely — but it materially reduces exposure for organizations that cannot redeploy immediately.

Model artifact controls should be applied at the network edge. Egress filtering from inference and retrieval hosts to model registries (HuggingFace, ModelScope, and similar) is a high-leverage control because it transforms a remote-execution vulnerability into a denial-of-service event for the exploit. Where business requirements demand model downloads from those registries, route them through an explicit proxy that can scan model artifacts and enforce allowlists at the repository level.

Network segmentation should reflect the actual sensitivity of vector store contents. RAG vector databases routinely contain the most sensitive operational documents in an organization; they should sit in network segments with the same egress controls applied to secrets management systems, not in the open developer network where they often originate. Identity-aware network access — short-lived service tokens, mutual TLS, and reverse-proxy enforcement — should be the default for any AI infrastructure component that holds embeddings derived from sensitive source material.

Strategic Considerations

The disclosure dynamics of CVE-2026-45829 — credible vulnerability reports, no maintainer response, public release of details ahead of an authoritative fix — are a structural feature of the current AI tooling ecosystem rather than a property of any one project. Enterprises building production AI on open-source infrastructure should plan for the case in which the upstream project is non-responsive at the moment of disclosure. That planning has implications for procurement (vendor due diligence should examine the project's documented response to past vulnerability reports), for architecture (compensating controls must be available without upstream cooperation), and for operations (incident response runbooks should treat disclosure-without-patch as a routine scenario for AI infrastructure rather than an exception).

The structural pattern across 2026 disclosures — Ollama, vLLM, MCP, ChromaDB — also reinforces the case for treating AI infrastructure as a distinct asset class with its own inventory, monitoring, and patching cadence. The "AI Vulnerability Storm" briefing observed that AI endpoints appearing in scan indices are now exploited on timelines measured in hours rather than days, and the same dynamic applies to vector

databases [7]. Asset inventories that surface ChromaDB, Pinecone, Weaviate, Milvus, Qdrant, and similar components, with their network reachability and authentication posture explicitly recorded, are now a baseline operational capability.

Finally, AI infrastructure security warrants formal inclusion in the vendor and architecture review process at the time of design rather than as a post-deployment audit. The pattern of exposure findings reported through 2025 and 2026 suggests that visibility latency — the gap between an AI component entering production and a security team gaining awareness of it — has been a contributing factor [7]. Pre-production architectural review — with explicit treatment of authentication, network exposure, and remote artifact loading — is almost always less expensive than the forensic triage that follows a confirmed compromise, particularly when cloud credentials must be rotated and embedded data must be treated as exfiltrated.

CSA Resource Alignment

This research note aligns with several active CSA frameworks and publications that address the specific gaps the ChromaToast disclosure exposes.

CSA Zero Trust Architecture for LLM Environments, released in March 2026, identifies the vector database as a distinct protect surface — separate from the AI API endpoint, the prompt interface, and the underlying cloud infrastructure — with its own access control, data flow, and monitoring requirements [10]. The architectural guidance that authentication should terminate at a controlled boundary before any code path executes maps directly to the recommended ChromaToast mitigations: placing the Python FastAPI server behind a reverse proxy is precisely the protect-surface boundary the Zero Trust guidance prescribes.

MAESTRO, CSA's agentic AI threat modeling framework, provides structured analysis across the seven layers of agentic AI architecture, including the infrastructure and data layers where ChromaDB resides [11]. MAESTRO's treatment of boundary collapse between layers — where authentication assumptions for one layer leak into the implementation of another — describes the design defect at the heart of CVE-2026-45829, where the data layer accepted configuration that triggered code execution at the runtime layer without traversing the identity layer in between.

CSA AI Controls Matrix (AICM) provides controls across 18 domains relevant to AI systems, including controls for authentication and access management, supply chain security, and monitoring. The AICM's Application Provider and Orchestrated Service Provider implementation guidelines establish expectations that map to ChromaToast remediation: explicit authentication before any privileged operation, restriction of remote artifact loading, and continuous monitoring for unauthorized

configuration changes. Organizations using AICM to assess vector database deployments should verify these specific controls during third-party AI vendor assessments and during internal architecture reviews [12].

CSA "AI Vulnerability Storm" Briefing, published in April 2026 with SANS, [un]prompted, and the OWASP GenAI Security Project, characterizes the operating environment in which CVE-2026-45829 disclosure occurred — AI-accelerated vulnerability discovery compressing exploitation timelines, traditional patch SLAs unable to keep pace, and supply-chain-style trust failures across the AI tooling ecosystem [7]. Its recommendations on machine-speed vulnerability response and on treating AI infrastructure as a priority patching surface are directly applicable to defensive planning around ChromaToast. Defenders working through the ChromaToast response should treat that briefing's guidance on network exposure, dependency posture, and identity architecture as directly applicable to the vector database tier.

References

[1] HiddenLayer. "ChromaToast Served Pre-Auth." HiddenLayer Research, May 2026.

[2] NIST. "CVE-2026-45829 Detail." National Vulnerability Database, May 18, 2026.

[3] Hadrian. "CVE-2026-45829 – ChromaDB Python server hands you RCE before it asks who you are." Hadrian Blog, May 2026.

[4] BleepingComputer. "Max-severity flaw in ChromaDB for AI apps allows server hijacking." BleepingComputer, May 19, 2026.

[5] UpGuard. "Open Chroma Databases: A New Attack Surface for AI Apps." UpGuard Research, 2025.

[6] SecurityWeek. "Unpatched ChromaDB Vulnerability Can Lead to Server Takeover." SecurityWeek, May 2026.

[7] Cloud Security Alliance, SANS Institute, [un]prompted, OWASP GenAI Security Project. "The 'AI Vulnerability Storm': Building a 'Mythos-Ready' Security Program." CSA, April 2026.

[8] The Hacker News. "We Scanned 1 Million Exposed AI Services. Here's How Bad the Security Actually Is." The Hacker News, May 2026.

[9] Sysdig Threat Research Team. "LLMjacking: From Emerging Threat to Black Market Reality." Sysdig, 2026.

[10] Cloud Security Alliance. "Using Zero Trust to Secure Data in LLM Environments." CSA Zero Trust Working Group, March 2026.

[11] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

[12] Cloud Security Alliance. "AI Controls Matrix (AICM)." CSA AI Controls Working Group, 2026.

[13] NIST. "CVE-2026-7482 Detail." National Vulnerability Database, 2026.

[14] NIST. "CVE-2024-50050 Detail." National Vulnerability Database, 2024.

[15] NIST. "CVE-2026-22778 Detail." National Vulnerability Database, 2026.

[16] NIST. "CVE-2025-23254 Detail." National Vulnerability Database, 2025.

[17] NIST. "CVE-2025-49596 Detail." National Vulnerability Database, 2025.