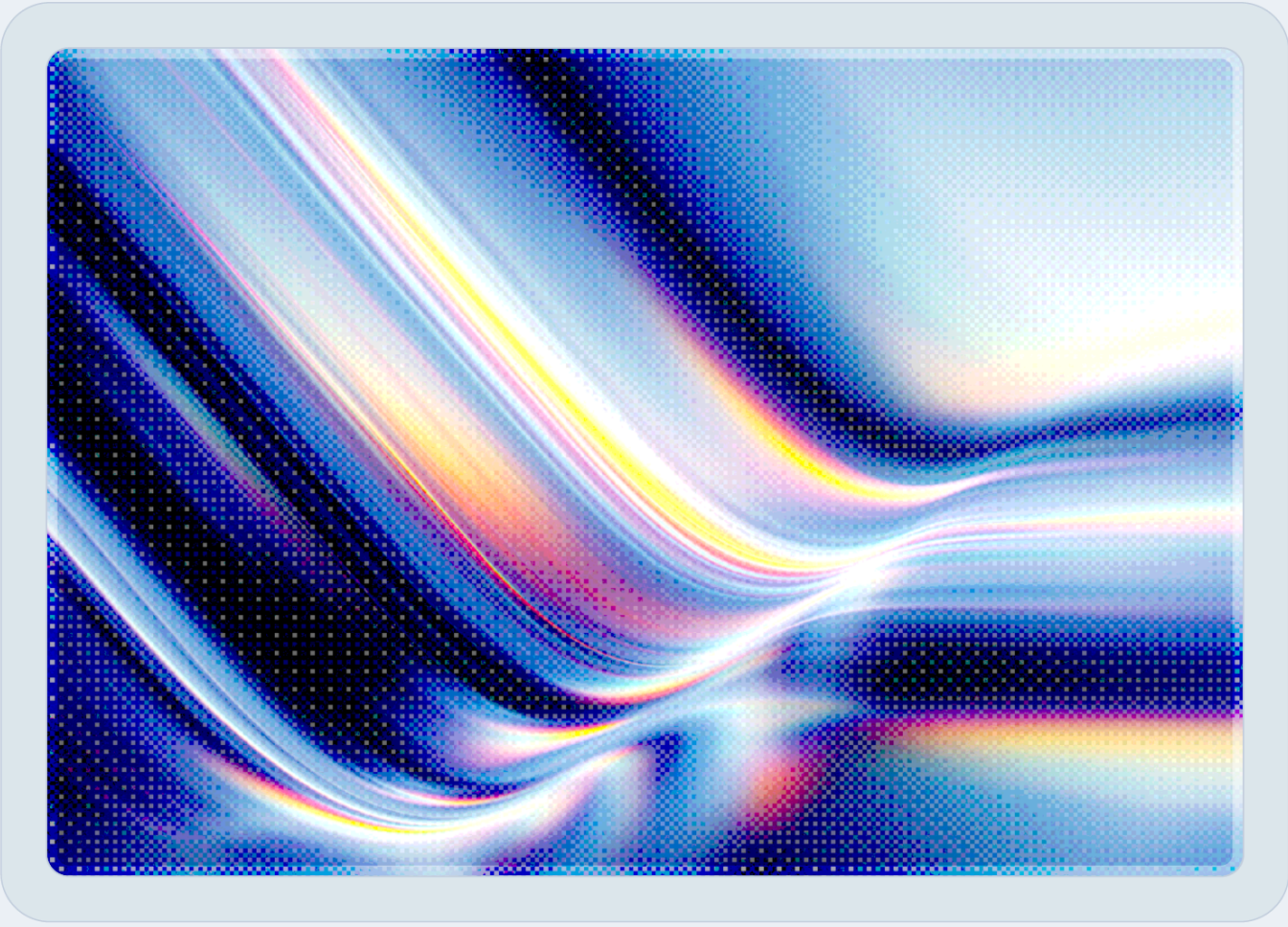
CSAI Foundation | Cloud Security Alliance

# Trojanized Offensive AI Tooling: The ARTEX Docker Backdoor

## Supply-Chain Risk in Red-Team Agent Frameworks: Security Implications and Guidance

2026-11-06

AI-assisted Rapid Research



CSAI cloud security alliance®

**© 2026 Cloud Security Alliance. Some rights reserved.**

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

*This document was generated with AI assistance and has not undergone official CSA review and approval processes.*

## Key Takeaways

A backdoored Docker Hub image of ARTEX, an open-source AI-driven penetration-testing platform, gave anyone who knew a secret trigger an unauthenticated root shell inside the ARTEX container. Depending on how the container was deployed, that access may extend to the underlying host. According to the researcher who disclosed it, the trojanized image `tomicdd/artex` was pulled roughly 9,000 times (a count of pulls, not unique deployments), and 62 internet-facing ARTEX consoles served the backdoored build between November 2 and 4, 2026 [1][2]. The backdoor was delivered without any visible change to application source. After the original developer withdrew ARTEX from GitHub, a fork of the project edited a single line in `docker-compose.yml` so that it pulled the malicious image [1][3].

The numbers alone understate the significance of the incident, because offensive AI consoles are unusually valuable to compromise. An ARTEX instance holds LLM API keys, client target lists, unpatched findings, captured credentials and sessions, and a network position inside or adjacent to the environments it tests [1][2]. A backdoor in such a tool hands an attacker the results of every engagement whose data the instance retains. It also lets the attacker steer the agent itself, which could turn an authorized assessment into an unauthorized intrusion [1][2].

The disclosure coincided with a November 4, 2026 CERT-In advisory, "Escalation of AI-Driven Attacks on BFSI Entities by Threat Actor 'CRAFTY FALCON'" [7]. As reported, the advisory names the "ARTEX family" alongside PentAGI, Cairn, and CyberStrikeAI as publicly available AI offensive frameworks used by an AI-assisted threat actor designated "Crafty Falcon" against Indian banking, financial services, and insurance (BFSI) organizations [5][6]. Public reporting does not connect Crafty Falcon to the backdoored image. The two share a toolkit, and no attribution between them has been demonstrated [2].

Organizations that run AI red-team agents should therefore treat them as privileged infrastructure. In practice, that means pinning images by digest, verifying provenance, keeping consoles off the public internet, separating secrets from agent workers, and rotating every credential an affected instance touched. Any deployment that used `tomicdd/artex` or `congdonglinux/artex` should be treated as fully compromised and rebuilt rather than cleaned [2][3].

# Background

ARTEX is an open-source, AI-driven autonomous penetration-testing system. The original upstream repository is no longer available, so this note draws its architecture description from a third-party source backup of the final ARTEX release [4]. That copy has not been vetted, and readers should not deploy from it. According to that backup, ARTEX is built as a Go single binary with an embedded Next.js frontend and a PostgreSQL backing store [4]. A planner agent generates attack "intents" and maintains a shared task list across wake-ups, and multiple worker agents execute those intents using Bash and HTTP tools. The design is described as a dual graph: an asset graph shared across tasks and a per-task exploration graph that tracks goals, facts, and findings. The backup also describes a man-in-the-middle proxy that records traffic, an approval queue for intercepting tool calls, and support for Anthropic and OpenAI models configured through environment variables or a web UI [4]. The container ships with tools such as `nmap` and `curl`, mounts skills from a local directory, and can connect to remote Model Context Protocol (MCP) servers. Authentication is JWT-based, and the administrator password is set on first visit to a `/setup` page [4].

The README in the backup repository restricts use to personal learning, code research, and local technical validation, and it disclaims use against any online system [4]. That restriction did not hold in practice. The Hacker News reported that the original developer removed ARTEX from GitHub after the tool appeared in real intrusions [1]. The removal left a gap that third-party mirrors and forks filled, and those copies became the delivery channel for the backdoor. The backup itself illustrates the pattern: it states that the original image, `autumn27/artex`, has been deleted and directs users to its own image, `mianhuatang11/artex`, instead [4]. One widely starred fork, reported with 2,449 stars and more than 600 forks, changed `docker-compose.yml` to pull the Docker Hub image `tomicdd/artex` [1][3]. Sources differ on whether this fork's repository name is `trhhubsd/ARTEX` or `trhhubsd/ARTEXX` [1][3], so defenders should check for both. The fork is reported to be no longer downloadable [1].

The backdoor was identified by an independent researcher using the alias Badbyte, who shared the findings with The Hacker News by email [1]. No independent primary publication by Badbyte was located at the time of writing, so the technical details below come from press and secondary analyses. According to that reporting, the trojanized build adds a hidden `verifyCaptcha()` routine to the login handler. If the CAPTCHA field begins with a secret marker, the remainder of the value is passed to `sh -c` and executed as root, and the output is returned to the caller before any credential check occurs [1][3]. The secret marker has not been published [3]. Badbyte observed that the routine "never shows up in a code review," because reviewers inspecting the fork's source would see legitimate ARTEX code. The malicious logic lived only in the prebuilt image [1].

The reported exposure timeline appears below. The push date is inferred from the reporting rather than confirmed.

| Date (2026) | Event | Source |
|---|---|---|
| ~October 8 (inferred) | Backdoored image reportedly pushed to Docker Hub; exact push date unconfirmed | [1][2] |
| October 20 | Earliest internet-facing console observed serving the backdoored build | [1][2] |
| November 2–4 | 62 backdoored consoles identified (26 U.S., 17 Hong Kong, 7 China, 12 unspecified) | [1][2] |
| November 4 | 1,645 hosts observed serving an ARTEX console of any build; CERT-In issues its Crafty Falcon advisory naming ARTEX among the actor's tooling | [1][2][5][7] |
| November 5 | Public disclosure via The Hacker News and follow-on analyses | [1][2][3] |

The 1,645 figure measures how widely ARTEX consoles are exposed to the internet, not how many were compromised [2]. Some of those hosts belong to legitimate researchers and red teams. Even so, 1,645 internet-reachable ARTEX consoles fit a pattern CSA has documented for self-hosted AI services more broadly [16].

The CERT-In advisory supplies the threat context. According to reporting on its contents, it updates an October 9 advisory on attacks against payment and disbursement APIs [5]. It describes an actor that selects targets with machine assistance, operates at machine speed, and drains funds by invoking victims' own payment and disbursement APIs from inside allow-listed infrastructure [5][6]. Reported techniques include abuse of AWS Systems Manager and Lambda, pivoting from exposed staging environments into production through shared credentials, file-upload and deserialization chains leading to PHP and Node.js web shells, Rs. 1 test transfers before larger thefts, and log deletion [1][5]. CERT-In attributes the activity to use of publicly available AI-driven offensive frameworks, including ARTEX, PentAGI, Cairn, and CyberStrikeAI [1][5][6]. CSA analyzed the advisory and the Crafty Falcon toolkit in a companion research note published a day earlier [19].

# Security Analysis

## Why Offensive AI Consoles Are High-Value Targets

Traditional penetration-testing tools are dangerous in the wrong hands. However, they generally do not centralize as much persistent engagement state as an autonomous agent platform that keeps everything in one database. An ARTEX instance stores asset graphs of target environments, findings that have not yet been remediated, credentials and session tokens captured during testing, recorded HTTP traffic, and the LLM provider keys that power the agents [1][4]. It typically runs with broad network reach so that it can scan and exploit, and the reported backdoor executes as root [1]. A single compromise therefore yields two prizes at once: the operator's infrastructure and a curated map of exploitable weaknesses in every client or internal system the operator has assessed.

The agentic design adds a third dimension. Badbyte told The Hacker News that an attacker holding the console can also quietly change what the agents do on subsequent runs [1][2]. That capability could plausibly be used to widen scope, plant persistence in target environments, or exfiltrate data under the cover of an authorized engagement, and such activity would be difficult to distinguish from legitimate testing in client logs. This is an inference from the architecture rather than a reported observation. Public reporting states that it is unknown whether the backdoor was actually exercised against the 62 consoles' owners or their targets [2][3].

## A Supply-Chain Pattern, Not a Code Vulnerability

The ARTEX case is a supply-chain substitution rather than a flaw in ARTEX itself. The attacker did not need to compromise the upstream maintainer or inject code into a reviewed repository. Once upstream disappeared, popular forks and the Docker Hub namespaces they referenced accumulated stars and pulls [2][3]. A one-line change to a compose file is easy to miss in review, and the standard `docker compose up -d` workflow does not show users which image digest a tag resolves to. Users who did not inspect the compose file would have had no indication of the substitution. The backup repository's README similarly directs users to pull a prebuilt image and to build from source only if the pull fails [4], a workflow that favors convenience over provenance.

Several conditions amplified the risk. Mutable tags let a namespace owner change image contents at will. Docker Hub namespaces are first-come and carry no inherent link to the upstream project. Removal of the canonical source also eliminated the reference point against which users could compare a copy. The table below summarizes the artifacts named in public reporting and their reported status as of November 5, 2026.

| Artifact | Type | Reported status |
|---|---|---|
| `tomicdd/artex` | Docker Hub image | Backdoored; ~9,000 pulls [1][3] |
| `congdonglinux/artex` | Docker Hub image | Backdoored; reported identical to `tomicdd/artex` [1][3] |
| `trhhubsd/ARTEX` / `trhhubsd/ARTEXX` | GitHub fork | Compose file pointed to `tomicdd/artex`; reported no longer downloadable [1][3] |
| `mianhuatang11/artex` | Docker Hub image | Image named by the `toandbd1` backup repo; reported to correspond to `mhtsec/ARTEX`; not reported backdoored, but unvetted [1][4] |
| `mhtsec/ARTEX` | GitHub copy | Linked to `mianhuatang11/artex`; reported no longer downloadable; unvetted [1] [2] |
| `RuoJi6/ARTEX` | GitHub copy | Unvetted; status not reported [1] |
| `autumn26`, `andy1099zz`, `dgddw`, `ylh440104` `/artex` | Docker Hub images | Unvetted; no public finding either way [1][3] |
| `autumn27/artex` | Docker Hub image | Named by the backup repo as the original upstream image, since deleted; may be distinct from `autumn26` [4] |

The absence of a backdoor finding for the unvetted entries should not be read as a clean bill of health. Public analysis has not cleared these artifacts, and the available reporting does not say whether they have been examined [2][3].

## Structural Weaknesses in Agentic Red-Team Frameworks

The ARTEX backdoor arrives shortly after a study found that many open-source offensive agent frameworks are structurally fragile. In "Red-Teaming the Agentic Red-Team," Pasquini et al. evaluated twelve open-source offensive agent systems, including PentAGI, CAI, PentestGPT, and STRIX; ARTEX was not among them [8]. In a controlled honeypot setting, they report that an attacker controlling the target could induce agents to download and run a staged trojanized binary in 97.8% of runs, aggregated across ten of the systems and six LLMs. Secrets such as LLM API keys and cross-session logs were reachable in 11 of 12 systems. Hard persistence was achieved on 10 of 12, host escape on 10 of 12, and full host remote code execution on 8 of 12 [8]. The authors attribute these outcomes to co-location of orchestrator and worker, shared writable mounts, excessive container capabilities, host networking, and exposed orchestrator APIs. They also find that guardrails enforced only at the orchestration layer do not constrain a compromised worker [8].

Taken together, the two findings describe a threat model with attack paths on both sides of the tool. On the inbound side, a malicious image or fork compromises the console before any engagement begins, as with ARTEX. On the target side, a hostile environment can compromise the agent during an engagement, as Pasquini et al. demonstrate. In both cases the attacker inherits an over-privileged, secret-laden system with network reach into sensitive environments. The ARTEX image is a concrete supply-chain instance of a risk that the academic work suggests is systemic across this tool category.

## Dual-Use Proliferation and Governance

CERT-In's naming of ARTEX, PentAGI, Cairn, and CyberStrikeAI in a financial-sector advisory is evidence that at least one financially motivated actor is using open-source AI offensive frameworks operationally [5][6]. That creates a governance problem for defenders as well as attackers. Security teams evaluating these frameworks for legitimate red-team work draw from the same distribution channels that threat actors use, sometimes after upstream maintainers have withdrawn their code. Practitioners who obtain these tools from community mirrors after upstream withdrawal risk running software of unknown provenance that holds their clients' sensitive data. The 62 backdoored ARTEX consoles show that this risk has already materialized for some operators. According to reporting, CERT-In also published AI security auditing guidelines alongside the advisory, requiring human approval for consequential agent actions, testing of agent permission boundaries and least privilege, and emergency interruption and credential revocation for high-risk systems [1][2]. Those controls apply as much to an organization's own offensive agents as to its production AI systems.

# Recommendations

## Immediate Actions

Organizations should first determine whether they are exposed. That means searching container registries, CI logs, Kubernetes manifests, compose files, and container runtimes for references to `tomicdd/artex` and `congdonglinux/artex`, and for any deployment cloned from the `trhhubsd` fork under either spelling [2][3]. Teams should also review login endpoint logs on any ARTEX console for anomalous or unusually long CAPTCHA values, which may indicate trigger attempts [3]. ARTEX may not log CAPTCHA contents, however, and an attacker with root access can delete logs, as CERT-In reports Crafty Falcon doing. The absence of such entries therefore does not show that the backdoor went unused.

Any host that ran an affected image should be isolated, taken off the internet, and rebuilt from known-good media rather than cleaned in place [2][3]. Every credential the console stored, captured, or could reach must be rotated, including LLM provider API keys, client credentials and session tokens gathered during engagements, database passwords, and cloud keys on the host. Findings produced after the image was pulled should be re-validated. Clients whose environments were assessed from an affected console should be notified so that they can review their own logs for activity outside the engagement window [2]. Teams that run any other ARTEX build should treat it as hostile until it has been verified against a trusted source [2].

## Short-Term Mitigations

Container provenance controls address the substitution mechanism most directly. Images should be pinned by immutable digest (`image@sha256:...`) rather than by mutable tags [9]. Where publishers provide signatures, teams should verify them with Sigstore `cosign` or Notary v2 (Notation) before deployment [11]. Docker Content Trust (Notary v1) is being retired, and its public service at notary.docker.io shuts down on December 8, 2026 [10], so new deployments should not rely on it. Where no signatures exist, teams should build from reviewed source in a controlled pipeline rather than pull community images. Before adopting any fork, they should diff compose files, Dockerfiles, and CI configuration against a known-good upstream commit, paying particular attention to `image:` lines [2]. Mirroring vetted images into an internal registry and blocking direct pulls of unvetted namespaces at the registry or admission-control layer closes the remaining gap.

Deployment hardening should follow. Offensive AI consoles should never be exposed directly to the internet; they belong behind a VPN or identity-aware proxy with strong authentication [2][3]. The application should run as a non-root user with dropped capabilities, granting only the specific capabilities that bundled tools require (for example, `NET_RAW` for `nmap` SYN scans). It should avoid host networking and should not share writable mounts between orchestrator and workers, since Pasquini et al. identify these conditions as enabling host escape [8]. LLM API keys and engagement credentials should stay out of worker containers and, where the framework permits, be issued as scoped, short-lived tokens through a broker.

## Strategic Considerations

Organizations should formally classify AI red-team agents as privileged security infrastructure, subject to the same procurement, change-management, and monitoring controls as a domain controller or a secrets vault. Third-party security firms that use such tools carry the same obligation to their clients. Engagement contracts should require disclosure of the agent frameworks in use, their provenance and patch status, and how captured client data is stored and destroyed.

Over the longer term, the community building these frameworks should adopt the compromise-oriented design principles proposed by Pasquini et al. Under that model, the worker is treated as malicious, holds no secrets, and is separated from the orchestrator by operating-system, network, and filesystem boundaries, with guardrails enforced below the application layer [8]. Publishers should provide signed images and build provenance aligned with SLSA so that consumers can verify where an artifact came from [12]. Registries and the community could also consider a mechanism for signed tombstone notices: when a maintainer withdraws a project, a notice identifying the last legitimate artifact digest would help users avoid substitution attacks of the kind seen here. Such a mechanism depends on a cooperative and reachable maintainer, which may not hold when, as here, a project is withdrawn because of misuse. Finally, human-in-the-loop approval for consequential agent actions, as CERT-In's guidelines require, can limit what a hijacked worker does mid-engagement [2]. It protects against a console-level compromise like the ARTEX backdoor only if the approval mechanism runs outside the console's trust boundary, for example at a separate egress or credential broker. The ARTEX approval queue itself runs inside the compromised application.

# CSA Resource Alignment

The ARTEX incident connects to several strands of CSA's published research. The most direct is CSA's research note "Crafty Falcon: Weaponized Pentest Agents Hit the Financial Sector," which analyzes the CERT-In advisory and the toolkit that includes ARTEX [19]. This note complements that analysis. It shifts the focus from how threat actors use offensive AI frameworks to the risks those frameworks pose to the defenders who run them.

CSA's research note "ClawHub Under the Microscope: Agentic AI Supply Chain Risk" documented how community marketplaces for agent skills became a persistent malware distribution channel [13]. The ARTEX case extends that analysis from skill marketplaces to container registries. In both settings, trust in a popular community artifact substitutes for verification of its provenance, and an agent's broad privileges turn a single malicious artifact into full environment compromise.

The note "Mastra npm Takeover: 144 AI Packages Backdoored" analyzed a hijacked contributor account used to backdoor an AI agent framework's package scope [14]. ARTEX represents a different variant of the same pattern. Rather than taking over a legitimate namespace, the attacker exploited the vacuum left when upstream disappeared and redirected users through a fork. The digest pinning, provenance verification, and internal mirroring controls recommended in that note apply directly here.

"Carbonato: Telegram-Controlled AI Agent Hijacks Docker Hosts" described attackers repurposing an unmodified agent framework on compromised Docker hosts and prioritizing theft of AI API keys [15]. "36,769 Exposed AI Endpoints: A Self-Hosted Supply Chain Crisis" quantified the broader exposure of self-hosted AI services on the public internet [16]. The 1,645 internet-facing ARTEX consoles reported in this incident fit that documented pattern.

For threat modeling, CSA's MAESTRO framework provides a layered approach to agentic systems [17]. Its deployment-infrastructure layer covers the container substitution, while the orchestrator–worker separation failures discussed above fall under its agent-frameworks and agent-ecosystem layers. The AI Controls Matrix (AICM) v1.1 supplies the control baseline for these risks [18]. Its supply chain management, identity and access management, and threat and vulnerability management domains map to the provenance, credential-isolation, and exposure-management recommendations in this note. Organizations formalizing governance over their own offensive agents can draw on CSA's Agentic AI Red Teaming Guide, which addresses red teaming practice for agentic AI systems [20].

# References


[1] Lakshmanan, Ravie. "CERT-In Flags AI-Driven Attacks as Backdoored ARTEX Exposes 62 Consoles to Root Access." The Hacker News, November 5, 2026.

[2] AgentOffense Team. "Backdoored ARTEX Console Gave Root to Anyone Who Knew the Trigger." AgentOffense, November 5, 2026.

[3] InfraStack. "Backdoored ARTEX Docker Images (CMP-202611-3clfry)." InfraStack Live Campaigns, November 2026.

[4] toandbd1. "ARTEX: AI Agent for RedTeam (unvetted third-party source backup; do not deploy)." GitHub, accessed November 6, 2026.

[5] MediaNama. "CERT-In Flags an AI Threat Actor Targeting BFSI Payment APIs for Unauthorized Fund Transfers." MediaNama, November 5, 2026.

[6] Inc42. "CERT-In Flags AI-Driven Cyberattacks on Indian BFSI." Inc42, November 2026.

[7] CERT-In. "Escalation of AI-Driven Attacks on BFSI Entities by Threat Actor 'CRAFTY FALCON' (advisory announcement)." Indian Computer Emergency Response Team via X, November 4, 2026.

[8] Pasquini, Dario, Michal Bazyli, Taras Fedynyshyn, and Artem Sorokin. "Red-Teaming the Agentic Red-Team." arXiv:2606.24496, June 2026 (revised August 2026).

[9] Docker. "docker image pull: Pull an image by digest (immutable identifier)." Docker Documentation, accessed November 6, 2026.

[10] Docker. "Content Trust in Docker." Docker Documentation, accessed November 6, 2026.

[11] Sigstore. "Verifying Signatures with Cosign." Sigstore Documentation, accessed November 6, 2026.

[12] OpenSSF. "SLSA Specification v1.0." Supply-chain Levels for Software Artifacts, 2023.

[13] Cloud Security Alliance. "ClawHub Under the Microscope: Agentic AI Supply Chain Risk." CSA AI Safety Initiative, July 2026.

[14] Cloud Security Alliance. "Mastra npm Takeover: 144 AI Packages Backdoored." CSA AI Safety Initiative, June 2026.

[15] Cloud Security Alliance. "Carbonato: Telegram-Controlled AI Agent Hijacks Docker Hosts." CSA AI Safety Initiative, September 2026.

[16] Cloud Security Alliance. "36,769 Exposed AI Endpoints: A Self-Hosted Supply Chain Crisis." CSA AI Safety Initiative, September 2026.

[17] Cloud Security Alliance. "MAESTRO: Agentic AI Threat Modeling Framework." CSA, accessed November 6, 2026.

[18] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2026.

[19] Cloud Security Alliance. "Crafty Falcon: Weaponized Pentest Agents Hit the Financial Sector." CSA AI Safety Initiative, November 5, 2026.

[20] Cloud Security Alliance. "Agentic AI Red Teaming Guide." Cloud Security Alliance, accessed November 6, 2026.