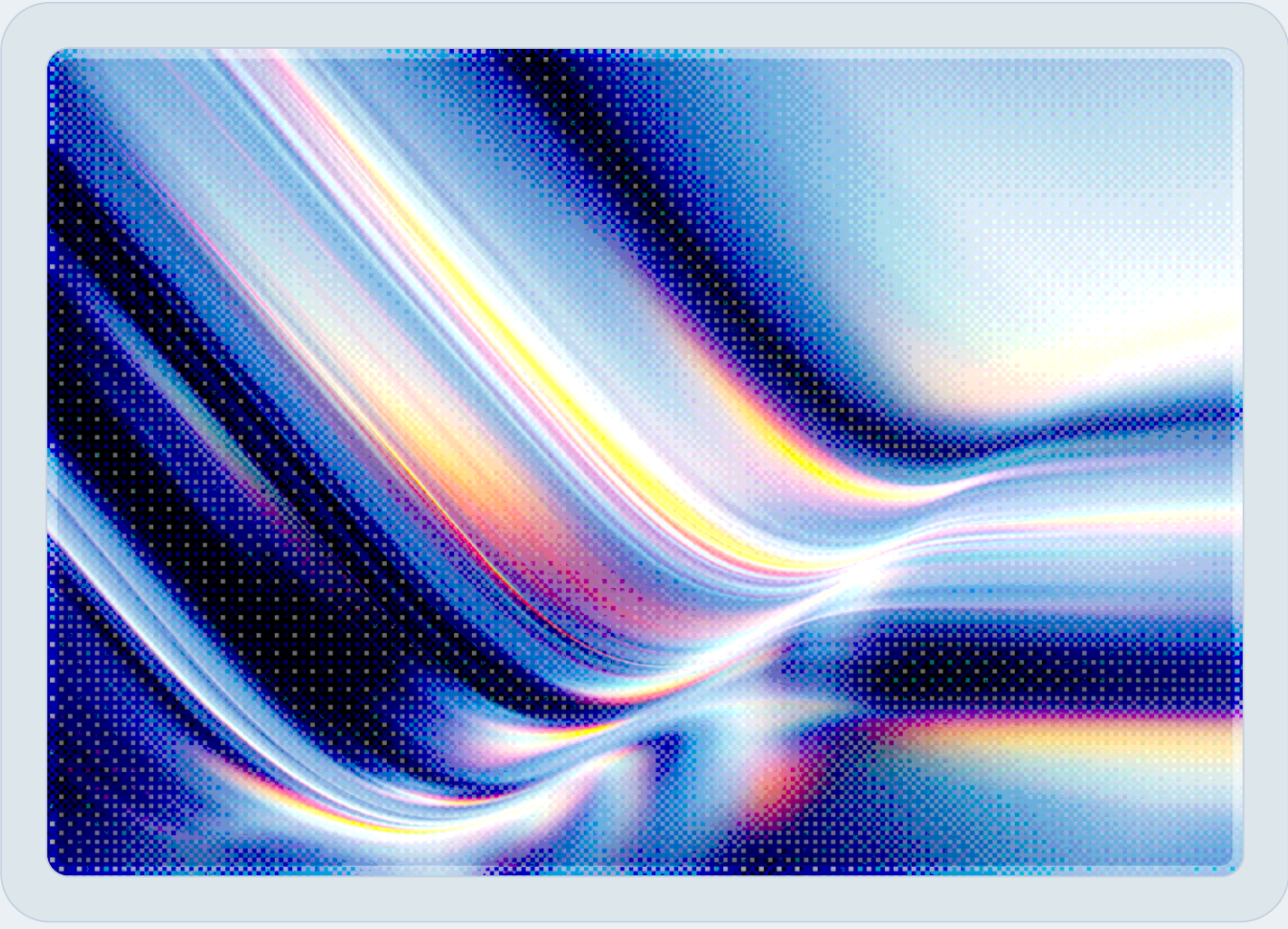
CSAI Foundation | Cloud Security Alliance

Fingerprinting Agentic Offensive Frameworks: Intent IDs, Cadence, and Self-Labels

Turning harness leaks into corroborating detection signals for ARTEX, Cairn, and related tooling

2026-11-27

AI-assisted Rapid Research



CSAI

CSA 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

Agentic offensive frameworks such as ARTEX and Cairn leave artifacts in victim environments that reflect how the harness organizes its own work rather than what the attacker intends. Taken together, published research from Wiz, Google Threat Intelligence Group (GTIG) and Mandiant, Elastic Security Labs, and the UK National Cyber Security Centre (NCSC) between October and November 2026 documents three families of these artifacts: incrementing "intent IDs" that the model copies into resource names, a characteristic machine cadence of compressed bursts separated by reasoning pauses, and self-applied labels such as `artex-`, `pentest-`, and `csai-` prefixes in user agents, accounts, and cloud resources [1][2][3][4][6].

Intent IDs (for example `i8920`, `i37257`, and `i001`) originate in ARTEX's shared database counter and Cairn's per-project counters, and they surface in role session names, SSM command comments, bucket and IAM names, CI workflow filenames, and test account names [1]. Elastic has proposed, but not yet merged, an endpoint rule keyed to them [5]. Cadence signals are similarly measurable: GTIG observed ARTEX execution sequences in bursts of roughly 2 to 25 seconds, and Elastic reports that CyberStrikeAI's implant checks in every five seconds with no jitter [3][4]. Self-labels, by contrast, come in two forms, framework-default strings such as user agents that change with releases and names the model chooses at run time that can vary from run to run. Both are perishable, but they currently appear often enough that Wiz and Elastic have published detection and hunting content keyed to them [2][4].

None of these signals is attribution. Wiz describes intent ID pacing as "a corroborating signal, not standalone attribution" and characterizes matches as "useful tool markers and pacing clues, but not operator fingerprints," because each installation keeps its own counter, unrelated installations reuse the same low number ranges, and one operator may run several installations in parallel [1]. Organizations should therefore add intent ID and self-label patterns to cloud audit and endpoint hunts now, pair them with anchoring context to suppress false positives, and shift rate limiting from IP addresses to accounts, tokens, and sessions as the NCSC recommends [6].

Background

Since September 2026, defenders have documented a marked rise in intrusions driven by open-source agentic penetration-testing frameworks repurposed for unauthorized use [6]. ARTEX is the most prominent of these. Microsoft reported that the China-aligned actor it tracks as Storm-0940 used ARTEX as early as September 2026, attempted sign-ins to more than 2,250 tenants, and in multiple cases moved from initial targeting to privileged access in less than three hours [7]. GTIG and Mandiant published a cross-framework analysis on October 30, 2026 covering ARTEX, HexStrike, Cairn, and PentAGI [3], and Wiz has since linked the "Black October" series of AI developer platform breaches to this class of tooling, reporting that an additional twenty related incidents were disclosed by vendors after its initial post [2]. The NCSC stated on November 24, 2026 that it "has seen a marked rise in AI-assisted cyber attacks" since September [6].

Many of these frameworks share a planner-worker architecture that matters for detection. A planner decomposes an objective into sub-goals, dispatches workers (usually LLM-driven agents with shell, HTTP, and cloud SDK tools), and records results in a shared store. Elastic's source-code review of five frameworks found that none enforces authorization in code by default, and three instead assert in their prompts that the activity is authorized [4]. NorForge's analysis of more than 20 exposed Cairn panels described a model of "cycles of intents and facts" posted to a shared blackboard, and found that projects that reached their goal took a little under 23 hours on average from start to goal [8]. In the ARTEX and Cairn families in particular, the attack is managed as a structured graph of numbered tasks, and that bookkeeping leaks [1][8].

Wiz coined the term "harness leaks" for internal framework details that end up in victim logs [1]. The concept is not entirely new: Anthropic's November 2025 disclosure of an AI-orchestrated espionage campaign, which CSA's analysis tracks as GTG-1002 [13], noted that the operation generated "thousands of requests, often multiple per second," a tempo that is itself a fingerprint of machine operation [10]. What has changed in late 2026 is the scale of the activity and the availability of the framework source code, which has allowed researchers to explain exactly where these artifacts come from and to build detection content against them. ARTEX's public repository has accumulated hundreds of forks, and Wiz, citing Censys data, reports more than 1,400 exposed ARTEX consoles in early November [2], so a single set of fingerprints potentially covers a large population of deployments.

Security Analysis

The three signal families differ in specificity, durability, and the telemetry needed to observe them. Intent IDs are highly specific to two framework families but trivially changed by a developer; cadence is observable across many frameworks but shared with legitimate automation; and self-labels are easy to match but perishable. The subsections below examine each family in turn and then consider how they can be combined into detections that are more reliable than any single indicator.

Intent IDs as Harness Leaks

The most distinctive signal is the intent ID. Wiz reviewed more than one hundred open-source offensive harness repositories and found two families, ARTEX (with forks such as Pojun and AntiFox) and Cairn (with descendants such as `sapper-v2`), that number every attack sub-goal and pass that number into the model's prompt and working directory [1]. Because the model sees the identifier repeatedly, it reuses it when naming the things it creates: files, accounts, GitHub workflows, cloud resources, and User-Agent strings. Wiz traces the ARTEX behavior to `agent/worker.go`, which places the identifier in a working directory of the form `tasks/<task>/i<N>`, a session name of the form `exp<N>-worker-i<N>`, and a prompt field labeled "intent id" [1]. The model is not instructed to emit these values to the target; it appears to reuse them as a readily available naming token because they recur throughout its context [1].

The two families number their intents differently, and the difference determines what a defender can infer. ARTEX draws intents, facts, findings, hints, and digests from a single PostgreSQL `BIGSERIAL` counter that by default never resets, so intent numbers climb across runs and targets and are rarely consecutive, although short consecutive runs can appear within a single burst of activity [1]. Cairn zero-pads identifiers to three digits and resets them for every project, producing small, gap-free sequences; Wiz examined one exposed Cairn host on which a single project held every intent from `i001` through `i292` [1]. Disclosed incidents illustrate how these values appear in practice, as summarized below.

Observed artifact	Location in victim environment	Framework family	Source
`i19ghci.yml`	GitHub Actions workflow filename (Barracuda disclosure)	ARTEX	[1]
`i3694canary`	Created account name (Bitrise disclosure)	ARTEX	[1]

Observed artifact	Location in victim environment	Framework family	Source
`artex-i37257`	Attacker artifacts (Chainstack disclosure)	ARTEX	[1]
`i63c2d.py`	Persistence poller (Ardor disclosure)	ARTEX	[1]
`artex-verify-i8920` to `i8923`	User-Agent strings	ARTEX	[1]
`pentest-iNNNNNN`	Workspace, pod, and counter names (ARTEX-Harvester)	ARTEX	[2]
`i001` to `i292`	Project graph on exposed Cairn host	Cairn	[1]

Because ARTEX counters are monotonic per installation, intent IDs support analysis beyond simple matching. Wiz describes how plotting IDs against timestamps yields a pacing signal, how a steady climb across rotating proxy IPs or across separate victims suggests a single installation, and how a backward jump (for example `i37257` on one day and `i4100` the next) indicates that the activity is not one continuous sequence [1]. In one case, Wiz found 36 intent IDs on the AWS side of an intrusion and 34 on the Azure side, all from one climbing sequence, which tied stolen AWS IAM users and an Azure service principal to the same run [1]. A jump in IDs much larger than the activity visible in one victim's logs may indicate that the installation is working other targets in parallel, although because facts, findings, and digests consume the same counter, some gap is expected even within a single target. Wiz used climbing IDs, together with a distinctive boto3 build and overlapping proxy pools, to link the ARTEX-Harvester campaign across more than a dozen organizations, at varying confidence [1][2].

The limits are significant, and the original researchers state them directly. Intent ID sequences distinguish installations, not operators: one operator can run several installations in parallel, every fresh deployment passes through the same low number ranges, and rebuilding the container or wiping the database breaks the sequence [1]. Cairn's per-project reset means cross-incident pacing works only for the ARTEX family [1]. Wiz characterizes matches as "useful tool markers and pacing clues, but not operator fingerprints" [1]. These caveats suggest that intent IDs are best used to cluster and scope activity once other evidence has established malicious intent, rather than as a primary trigger.

Endpoint Detection of Intent IDs

Elastic has proposed an endpoint rule that operationalizes the intent ID signal. The pull request, opened November 26, 2026 and still awaiting approval at the time of writing, defines the identifier as the letter `i` followed by four to seven digits and a short filename token, with examples such as `i42463_dbg.b64` and `i43345sys` [5]. The rule applies a regular expression to process command lines, executables, and file paths in Elastic Defend behavior alerts and fires when a host accumulates ten or more such alerts in an hour, or alerts from at least two distinct behavior rules [5]. The design choice is instructive: a single `i<digits>` match is too common to alert on, so the rule uses the identifier to escalate hosts that are already generating behavioral alerts.

Elastic deliberately excludes two- and three-digit forms because of collisions with `i18n` and `i386`, and lists false-positive sources including short tokens inside base64 blobs and build identifiers [5]. Teams building their own variants should expect similar tuning work. The four-digit floor means Cairn's three-digit identifiers fall outside this particular rule, and it also misses low-numbered intents from newly deployed ARTEX installations, such as the `i19` and `i63` artifacts in the table above. Cloud-side hunts that pair short `i<N>` tokens with anchors such as `artex-`, `pentest-`, or `tasks/<task>/i<N>` paths can cover that range.

Machine Cadence

The second signal family is tempo. Agentic frameworks alternate between rapid tool execution and pauses while the model reasons, producing a rhythm that may differ from both human operators and conventional scanners. GTIG reported that while intrusions showed extended pauses between operational phases, "individual ARTEX execution sequences ran in compressed bursts of roughly 2 to 25 seconds," and in one operation observed 27 events in 23 seconds of active burst time while probing AWS permissions [3]. The NCSC describes the same pattern qualitatively as "bursts of requests followed by short thinking phases," accompanied by a wide variety of endpoints and parameters and increased error rates [6].

Cadence also appears at larger volumes and in command-and-control channels. Microsoft observed a Storm-0940 password spray, part of its automated ARTEX campaign, peaking at 2,900 failed logons per minute against 327 accounts across eight organizations [7]. Abstract Security reports that spray campaigns associated with these frameworks run with a median gap of about one second between attempts for 20 to 60 minutes, and separately notes that these frameworks rotate client identities such as user agents [9]. Raw spray rates alone do not distinguish agentic tooling from conventional spray tools; the distinctive element is the burst-and-pause structure and the rotation of client identities. Elastic

notes that CyberStrikeAI's implant "checks in every five seconds with no jitter," a regularity that beaconing analytics are designed to catch [4]. The table below summarizes the documented tempo values, with campaign duration included as context rather than as a detectable signature.

Cadence signature	Reported value	Source
ARTEX execution bursts	Roughly 2 to 25 seconds, with extended pauses between phases	[3]
Single ARTEX burst	27 events in 23 seconds of active time	[3]
Storm-0940 password spray (ARTEX)	Peak of 2,900 failures per minute	[7]
Framework-driven spray campaigns	Median gap of about 1 second for 20 to 60 minutes	[9]
CyberStrikeAI implant beacon	Every 5 seconds, no jitter	[4]
Cairn campaign timeline (context)	A little under 23 hours on average for projects that reached their goal	[8]

Cadence may prove more durable than naming, because the pauses between bursts partly reflect model inference time rather than a naming choice, although beacon intervals and request pacing can still be tuned by the operator. It is also less specific. Legitimate automation, CI pipelines, and sanctioned AI agents produce bursty traffic as well. For this reason, Elastic's published hunt combines tempo with diversity, alerting when a single source IP in a one-minute window issues at least 20 requests spanning at least three user-agent tool families (for example, AWS SDK, Python, curl, and ARTEX's `norma` page reader) [4]. Because these agents "routinely rotate through proxies and VPNs," the NCSC recommends applying rate limits to accounts, API tokens, and authenticated sessions rather than IP addresses [6]; CSA recommends keying tempo analytics to the same identities for the same reason.

Self-Labeling

The third signal is the tendency of these agents to label their own work. Wiz reports that opportunistic agents "label their own work: `pentest`, `probe`, `recon`, and incrementing task numbers" in user agents, resource names, and usernames [2]. In the ARTEX-Harvester campaign, this took the form of `pentest-recon` user agents and `pentest-*` workspaces and pods; in a CyberStrikeAI intrusion,

nearly every resource the attacker created carried a `csai-` prefix, including `csai-escape` pods and a `csai-pentest-role` IAM role [2]. GTIG documented ARTEX files named `artex-probe-*` with five-digit, roughly incrementing numbers, and created accounts whose plaintext passwords included `ArtexProbe`, a detail visible to defenders who log the command lines used to create accounts [3]. Elastic catalogs framework-default user agents such as `artex-enrich/1.0`, `norma/0.4`, `CyberStrikeAI-WebShell/1.0`, and `HexStrike-HTTP-Framework/1.0` [4].

These examples fall into two distinct categories that decay differently. Framework-default strings, such as the user agents Elastic catalogs, are hard-coded and change with releases, while names the model chooses at run time, such as `pentest-` or `csai-` resource prefixes, can vary from run to run. Self-labeling of the second kind plausibly follows from the same prompt design that produces intent IDs. Frameworks that tell the model it is conducting an authorized penetration test, as Elastic documents for ARTEX, CyberStrikeAI, and PentAGI [4], may encourage the model to name artifacts the way a legitimate tester would, so that they can be identified and cleaned up afterward. This is an inference rather than a documented mechanism, but it is consistent with NorForge's finding that some Cairn projects "start out claiming to be authorized assessments," an observation drawn from a panel NorForge rates as malicious with low confidence [8]. Whatever the cause, both kinds of label are easy to change. Elastic warns that framework defaults "can change in the next release," and Wiz describes its own rules for this activity as "expected to be perishable" [2][4].

Defenders adopting these hunts should also note that some labels collide with legitimate naming. The `pentest-` prefix is a natural choice for sanctioned testing, and `csai-` is a plausible internal prefix at organizations that use the abbreviation, including the CSAI Foundation itself. Matches on such prefixes should be checked against an allow-list of known internal resources and authorized engagements before escalation.

Combining Signals

Each signal on its own produces either false positives (generic `i<digits>` tokens, bursty automation, the word "probe") or brittle true positives (exact user-agent strings). The published guidance consistently recommends pairing them. Wiz advises against alerting on an `i<N>` pattern alone and instead pairing it with an anchor such as an `artex-`, `pentest-`, `probe`, or `canary` prefix, or a leaked `tasks/<task>/i<N>` path [1]. Elastic's proposed endpoint rule pairs intent IDs with existing behavioral alerts [5], and its network hunt pairs tempo with user-agent diversity [4]. Wiz also notes that names survive in error messages for denied cloud API calls [1], which means that even failed reconnaissance can leave intent IDs and self-labels in provider audit logs. The NCSC asks whether

defenders can correlate across log sources to detect these signals [6], and correlating them across web, identity, cloud control plane, and endpoint telemetry is likely to produce more durable detection than any single indicator.

Recommendations

The recommendations below are ordered by time horizon. Immediate actions draw directly on published indicators and detection logic, short-term mitigations build analytic and preventive capability over the coming quarter, and strategic considerations address how detection investment should evolve as framework developers adapt.

Immediate Actions

Security teams should run retrospective hunts in cloud audit logs (CloudTrail, Azure Activity and Entra logs, GCP Audit Logs) for intent ID patterns and self-labels in the fields Wiz identifies: role session names, SSM command comments, bucket, role, user, and policy names, tags and descriptions, KMS aliases, security group names, Azure run-command names, VM computer names, and app registration names [1]. Hunts should include denied API calls, since names persist in error messages [1]. On endpoints, teams running Elastic can evaluate the proposed intent ID rule [5], and others can adapt its logic by matching `i` plus four to seven digits plus a short token in command lines and file paths and escalating only hosts that also carry behavioral alerts. Because that pattern misses Cairn identifiers and low-numbered ARTEX intents, cloud-side hunts should also search for shorter `i<N>` tokens paired with anchoring prefixes or paths. Web and API logs should be searched for the self-identifying user agents documented by Elastic and Wiz [2][4]. Once a match is confirmed against anchoring context, response should include credential and session revocation, not only IP blocking: Elastic observes that CyberStrikeAI switches proxies and user agents once blocked and that the ARTEX actor fell back to open proxies [4].

Short-Term Mitigations

Over the next quarter, teams should build tempo analytics keyed to identities and sessions rather than source addresses, extending the NCSC's rate-limiting guidance [6], and should implement correlated hunts similar to Elastic's tempo-plus-diversity query [4]. Where intent IDs are found, analysts should record the values with timestamps and source infrastructure so that ARTEX-family pacing can be used to scope incidents and identify parallel activity, keeping in mind that this distinguishes installations rather than operators [1]. Rate limits on account creation, minimal error messages in application responses, and decoys such as honeytokens can raise the cost of the exploratory behavior these frameworks depend on

[6]; limiting application error detail does not reduce the value of provider audit logs, which continue to record denied calls. Detection content keyed to exact strings should be tagged with an expiry or review date, since both Wiz and Elastic expect these indicators to change [2][4].

Strategic Considerations

The longer-term lesson is that harness design leaks into the environment, and defenders can study open-source offensive frameworks the way they study malware families. Organizations with threat intelligence capacity should track framework repositories and forks for changes to identifier schemes, prompts, and default user agents. At the same time, intent IDs and self-labels reflect current design choices that a more careful developer could remove, while the burst-and-pause rhythm and error-driven exploration appear to reflect deeper properties of model-driven operation, though operators can tune pacing to some degree. We judge that investment should therefore weight durable behavioral detection, including identity velocity monitoring, control-plane anomaly detection, and cross-source correlation, above indicator matching. Organizations that run sanctioned AI agents should also ensure those agents are identifiable through registered identities and consistent labeling, so that legitimate agent traffic can be separated from hostile agent traffic that shares the same tempo.

CSA Resource Alignment

This note extends a series of CSA AI Safety Initiative analyses of the 2026 agentic intrusion wave. *ARTEX and Its Forks: Agentic Pentest Frameworks as Attack Infrastructure* [11] describes how ARTEX and its fork ecosystem became operational attack infrastructure, and *AntiFox in the Wild: Forkable Agentic Frameworks Reach AWS Secrets* [18] examines one of the ARTEX forks named above; the fingerprints discussed here give practitioners concrete ways to detect that infrastructure at work in their own environments. CSA's research note *Fifty Dollars per Breach: Detecting Autonomous Agent Intrusion Campaigns* [17] covers the second framework family whose intent ID scheme is analyzed in this note. *Black October: AI Developer Platform Breaches and Industrialized LLMjacking* [12] covers the AI developer platform breach cluster, including Zeabur and Chainstack, that Wiz later tied to agentic offensive tooling. It should be read alongside the indicator table above, which draws on several disclosures from the same period.

LLM Agents as Offensive Post-Exploitation Tools [13], CSA's analysis of the GTG-1002 campaign, highlighted machine-speed reconnaissance and non-human identity velocity as detection priorities. The cadence findings here are consistent with that framing and add specific burst and beacon measurements from late-2026 frameworks. *Autonomous Agentic AI Adversaries* [16] provides broader context on autonomous adversaries and defensive detection. For threat modeling, MAESTRO [14] provides a layered

approach to agentic systems that applies equally to offensive frameworks. Its Agent Frameworks layer is where intent ID schemes originate, and its Evaluation and Observability layer is where defenders can capture the resulting leaks. Controls in the AI Controls Matrix v1.1 [15], particularly within the Logging and Monitoring, Identity and Access Management, and Threat and Vulnerability Management domains, give organizations a structure for building identity-keyed tempo monitoring and indicator hunting into their control programs.

References

[1] Wiz Research (R. McCarthy, H. Ramati). "Inside Agentic Offensive Frameworks: Intent IDs." Wiz Blog, November 25, 2026.

[2] Wiz Research (R. McCarthy, H. Ramati). "Agentic Attacks: In The Wild." Wiz Blog, November 23, 2026.

[3] Google Threat Intelligence Group and Mandiant. "ARTEX and Friends: Enter Agentic Offensive Frameworks." Google Cloud Blog, October 30, 2026.

[4] M. Ayenson, Elastic Security Labs. "Inside the Loop: Five AI Attack Frameworks That Never Ask Permission." Elastic Security Labs, November 13, 2026.

[5] Elastic. "New Rule: Multiple Elastic Defend Alerts with Agentic Intent IDs, Pull Request #7251." GitHub, elastic/detection-rules, November 26, 2026.

[6] National Cyber Security Centre. "Familiar Weaknesses, Exploited Faster: Defending Against Agentic Cyber Attacks." NCSC Blog, November 24, 2026.

[7] Microsoft Threat Intelligence. "AI-Driven, Large-Scale Nation-State Campaign Using ARTEX Enables Supply Chain Compromise." Microsoft Security Blog, October 21, 2026.

[8] NorForge. "Inside Cairn: Open Windows into Massive AI-Driven Cyber Operations." NorForge Blog, October 19, 2026.

[9] Abstract Security Threat Research Organization. "Countering Threats in the Era of Agentic Attacks." Abstract Security Blog, November 10, 2026.

[10] Anthropic. "Disrupting the First Reported AI-Orchestrated Cyber Espionage Campaign." Anthropic, November 2025.

[11] Cloud Security Alliance. "ARTEX and Its Forks: Agentic Pentest Frameworks as Attack Infrastructure." CSA AI Safety Initiative Lab Space, November 2026.

[12] Cloud Security Alliance. "Black October: AI Developer Platform Breaches and Industrialized LLMjacking." CSA AI Safety Initiative Lab Space, November 2026.

[13] Cloud Security Alliance. "LLM Agents as Offensive Post-Exploitation Tools." CSA AI Safety Initiative Lab Space, April 2026.

[14] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 6, 2025.

[15] Cloud Security Alliance. "AI Controls Matrix v1.1." Cloud Security Alliance, 2026.

[16] Cloud Security Alliance. "Autonomous Agentic AI Adversaries." CSA AI Safety Initiative Lab Space, June 2026.

[17] Cloud Security Alliance. "Fifty Dollars per Breach: Detecting Autonomous Agent Intrusion Campaigns." CSA AI Safety Initiative Lab Space, October 2026.

[18] Cloud Security Alliance. "AntiFox in the Wild: Forkable Agentic Frameworks Reach AWS Secrets." CSA AI Safety Initiative Lab Space, November 2026.