

CyberArmor Platform — Capability Definitions

68 capabilities: what each one means, the risk it addresses, where it runs, and its honest status

Audience: CISOs, security architects, risk and compliance leaders, and investors.

Date: 2026-10-03

How to read this. For every capability: a plain-English definition, a concrete use case, the risk it addresses, an honest status, and — where a limit exists — the limit, stated here rather than discovered later. **Teams** flags which group at a large enterprise typically owns the problem: **SEC** = cybersecurity, **RISK** = risk/compliance, **AI** = the business/AI-agent team, **ENG** = engineering/dev tooling.

Status legend (from our engineering status page, reconciled against the source repository): **Working** — runs end-to-end today, tested. **Configurable** — implemented; needs environment/config to activate. **Pilot** — functional; being exercised with design partners. **Roadmap** — not built yet.

On statuses. We publish maturity per capability rather than a single “available” claim, because the distinction between *working today*, *needs configuration*, *in pilot* and *not built* is the distinction a buyer actually needs. Where a capability has a known boundary — a format we do not parse, a surface that carries request context only — that boundary is written into its entry.

Part 1 · Prevent and enforce (1–19)

1 · Pre-ingestion trust gate — checked before the model reads it — Working · Teams: SEC, AI

A control point that evaluates a URL or piece of external content *before* any consumer — a human, a browser, an application, or an AI agent — ingests it. The gate canonicalizes the destination, checks reputation, fetches the content safely, scores it for phishing, hidden prompt injection, promptware, and IOCs, then returns a policy verdict: allow, warn, redact, sandbox, block, or isolate. This differs from every runtime prompt filter: those inspect content as the model is already consuming it; the gate decides *before exposure*, in a sealed environment. **Use case:** your internal Claude deployment or the customer-facing agent needs web context; a page carrying a hidden instruction is blocked or redacted before it ever enters model context. **Risk:** indirect prompt injection — the attack class where the payload arrives in *content*, not from the user.

2 · Sealed / sandboxed fetch of external content — Working (production requires network-isolated deployment, documented) · Teams: SEC

When the gate fetches a URL, it does so with no user credentials, no cookies, no session — GET-only, size- and time-capped, every redirect hop re-validated against SSRF rules. The content is examined in a sealed context that has nothing to steal. **Use case:** a link in a customer email or ops ticket is evaluated without the analyst’s session ever touching it. **Risk:** SSRF into internal services, credential replay, drive-by content executing in a privileged context.

3 · Sandboxed detonation of untrusted URLs in an isolated browser — Configurable (off by default — set `URL_TRUST_GATE_DETONATION_DEFAULT=on`; must run on an isolated network segment; Kubernetes NetworkPolicy hardening on roadmap) · Teams: SEC

For URLs that static fetching can’t judge, a one-shot Chromium instance (Playwright, on a dedicated network with no route to internal services) actually renders the page — exposing JavaScript-built DOM, CSS-hidden text (`display:none` payloads), and zero-width/Unicode-tag encoded instructions that never appear in raw HTML. **Use case:** deep-mode check of an unknown or suspicious URL before an AI agent is allowed to read it; forensic re-check after an analyst flags a miss. **Risk:** injection payloads that only exist after the page renders.

4 · Model-file static inspection without execution — Working on the endpoint agent · Teams: SEC, ENG

When a serialized model file (`.pkl`, `.pt`, `.ckpt`, `.joblib`, and similar) lands on a machine, the agent disassembles its pickle opcode stream — it never loads, unpickles, or executes the file — and reports the specific import that would give the file’s author

code execution on load. It recognizes how payloads *actually* serialize (e.g. `posix.system`, not `os.system`) and emulates the pickle memo so a payload can't hide behind a memoized string. `.safetensors` is deliberately not flagged — that format cannot carry code. **Known limits, stated up front:** it is an unsafe-import detector keyed on a denylist, not a proof of safety; and HDF5/Keras coverage is a narrow heuristic. (`.npz` archives are parsed, including a pickle hidden behind a NumPy header.) **Use case:** a quant downloads a checkpoint from Hugging Face; the verdict arrives the moment the file is written, on the workstation — not weekly, in a cloud registry. **Risk:** a poisoned checkpoint is arbitrary code execution the moment someone loads it.

5 • Dataset supply-chain scanning — `Working` (inventory) / artifact scanning as row 4 • Teams: SEC, ENG

Two halves: the AI Bill of Materials (A-BOM) inventories models and datasets across endpoints, code repositories, cloud sources (including e.g. BigQuery datasets and object storage), and artifact registries — so you can answer “what AI artifacts exist here and where did they come from”; and serialized data artifacts get the same static code-execution inspection as row 4. **Honest scope:** this is provenance inventory plus embedded-code detection — not statistical data-poisoning analysis of training sets. **Use case:** an examiner or internal audit asks which models and datasets are in use and their sources; the A-BOM answers from live inventory. **Risk:** unknown/untracked AI artifacts and code smuggled inside data files.

6 • Indirect prompt-injection / hidden-instruction detection — `Working` • Teams: SEC, AI

Detection of instructions hidden *inside content*: zero-width characters, homoglyphs, Unicode-tag encoding, CSS-hidden text, encoded segments — normalized and extracted before scoring, with an ML classifier plus a heuristic ensemble, and session-level correlation that recognizes multi-turn “promptware” attack chains rather than judging each message in isolation. **Use case:** a web page, resume, or document pulled into your internal Claude's context carries invisible “ignore your instructions” text; it is caught at ingestion. **Risk:** the model obeys content instead of the user — currently the top practical attack on deployed LLM applications.

7 • Runtime prompt guardrails / LLM firewall — `Working` • Teams: SEC, AI

The conventional layer of this category: inline screening of prompts and requests for injection, jailbreak attempts, toxicity, and sensitive data, at the proxy, gateway API, SDK, or RASP layer. This is table stakes for the category, and capabilities 1–6 are what sit *in front of* it. **Use case:** every prompt to an approved provider passes policy before it leaves. **Risk:** jailbreaks and malicious prompts reaching the model unscreened.

8 • Egress DLP for AI — 30-class catalog (27 structured + 3 NER) — `Working` • Teams: SEC, RISK

Data-loss prevention tuned for AI egress: 27 structured detector classes (SSN, credit card, bank routing, passport, driver's license, DOB, and a full secrets family — AWS/GCP keys, GitHub/Slack/Stripe/OpenAI/Anthropic tokens, JWTs, private keys) plus 3 NER classes (person, location, organization) plus semantic similarity detection. **Honest scope:** the catalog also carries IP-address, URL and crypto-address as NER-only classes, but emitting them needs a NER model whose label set includes them; the shipped default does not. Context-aware: it knows an email address in a login form is authentication, not a leak — so it doesn't redact your credentials into a broken login. Actions are graded: warn, redact (with optional per-tenant HMAC pseudonymization so auditors can correlate without recovering the value), or block. **Use case:** an employee pastes a customer letter containing an SSN into a chatbot; the SSN is redacted before the request leaves. **Risk:** NPI/PII/secrets in prompts — the #1 reason banks ban AI tools outright.

9 • PHI detection + clinical de-identification (HIPAA Safe Harbor) — `Pilot` (recently added) • Teams: RISK

Detection of protected health information classes and de-identification aligned to the HIPAA Safe Harbor method, backed by a HIPAA policy pack (22 controls). For a financial-services buyer this is not the core purchase — it demonstrates the regulated-vertical depth of the same DLP engine, of which the FINRA and SEC packs are the finance equivalents. For healthcare-adjacent workflows it is the core purchase. **Risk:** PHI in AI traffic for any healthcare-adjacent workflow.

10 • Model-response inspection, not just the prompt — `Working` on proxy surfaces • Teams: SEC, AI

Most tools only inspect what users send. This inspects what the model *returns*: command-execution patterns (`curl | bash`, `powershell -enc`), XSS/script payloads, browser-data exfil sinks, PII in output, and the dangerous-code-generation pattern generic classifiers miss (file-walk + encrypt + delete — ransomware shape). **Honest scope:** response inspection ships on both proxy surfaces; the browser extension, SDK, and RASP surfaces carry request context only today. **A streamed response is inspected but not blocked.** Since 2026-09-20 both proxies forward a `text/event-stream` answer to the client frame by frame, because holding it until the model finished froze every streaming AI client; a byte already delivered cannot be recalled, so those answers are scanned from a bounded retained copy and recorded as observed-not-enforced with a named coverage gap, never as

a clean or enforced scan. A non-streamed response is still blocked outright. The endpoint's own coverage report (capability 55) states how many responses were of each kind. **Use case:** your coding assistant returns a "fix" that quietly exfiltrates browser data; a buffered response is blocked before it renders, and a streamed one is flagged with the record saying it could not be stopped. **Risk:** the model as an attack vector *into* your environment.

11 · File-upload content inspection — `Working` · Teams: SEC, RISK

The hosted file-scanning API is production-deployed; the browser pre-submission upload gate is pilot-ready. Files being uploaded on covered paths are opened and inspected — not just fingerprinted by name/type — with a dedicated `block uploads` policy action. **PDF, DOCX, XLSX and PPTX are parsed at rest**, and images are read with **offline OCR** (no cloud service, so it holds in an air-gapped deployment). File type is sniffed from the bytes rather than trusted from the extension. **Verification update, 2026-10-03:** production TXT/XLSX content scanning and text redaction returned successful responses; browser/proxy regression tests verified replacement bytes and blocked uploads. Extension 2.3.7 holds covered submissions until scan and policy evaluation complete. **Honest scope:** XLSX can be inspected, but Excel workbook redaction/rewrite is not implemented; a redact decision blocks the attachment instead of submitting its original bytes. Native Codex/Claude protection requires a working, routed endpoint proxy. Live vendor upload workflows remain a client validation task. Offline OCR is intended for document scans; handwriting and photographs of screens are not validated. **Use case:** an analyst drags a client spreadsheet into a chatbot; content inspection decides redact/block, not the filename. **Risk:** bulk exfiltration via attachments — the highest-bandwidth leak path to AI services.

12 · Seven graded policy actions at the moment of use — `Working` · Teams: SEC

Enforcement is graded, not binary: allow · monitor · warn · redact · block-uploads · block · isolate (the trust gate additionally enforces sandbox). Every action is ranked server-side, so an action the server does not recognise can no longer fall through to "allow" — a real defect, fixed: six actions once enforced in Chrome and failed open on the endpoint agent, the proxy and the gateway. The policy builder only offers actions the selected surface can actually enforce, so an unenforceable rule can't be saved. **What `isolate` means here, precisely:** on the URL-trust-gate surfaces it enforces as a hard block. It does **not** cut the host off the network — automated host containment is a separate endpoint mechanism with its own triggers (capability 44), not something a policy action reaches today. **Use case:** start a new control in monitor, move to warn, then redact, then block as confidence grows — the rollout path a change-averse enterprise actually follows. **Risk:** binary block/allow tools get switched off after the first false positive; graded ones survive.

13 · Every verdict names the engine that produced it — `Working` (recent) · Teams: SEC

Every detection verdict carries attribution: which engine fired (ML classifier, heuristic ensemble, reputation feed, hidden-text extractor), with what score, and why. Analysts see "credential_harvest 0.85 — brand/domain mismatch + credential form," not an unexplained "blocked." **Honest scope:** engine-level attribution ships on detection-service verdicts (proxy, endpoint agent, gateway API). The URL trust gate's own verdict carries per-threat-class scores and IOC sources but not the engine name — propagating the detector into the gate response is open work. **Use case:** your SOC triages an alert in seconds and tunes the specific engine that's noisy instead of distrusting the whole product. **Risk:** unattributed verdicts are unactionable — and unactionable tools get ignored.

14 · One rulebook, centrally authored, enforced on every surface — `Working` · Teams: SEC, RISK

Policies are written once against a governed field registry and compiled to each enforcement surface — browser extension, proxies, endpoint agent, trust gate, gateway API. RASP is the exception: it enforces a local, in-process ruleset configured at the application, not the central rulebook — wiring it to policy sync is open work. Every policy field must have a real producer on the surface it targets (this is enforced by tests), so rules can't silently reference data a surface doesn't have. **Use case:** "no SSNs to unapproved AI providers" is one rule, enforced identically in the browser, the IDE, and server-side apps. **Risk:** per-surface rulebooks drift, and the gap between them is where incidents live.

15 · Tenant-set fail mode on the endpoint proxy, fail-closed by default — `Working` · Teams: SEC

When a detection or policy component is unreachable, *you* decide per tenant whether traffic fails closed (blocked until inspection is restored — the default) or fails open (availability first). It's a visible setting, not an undocumented behavior. **Honest scope:** the tenant setting is read by the endpoint local proxy. The containerized network proxy and the runtime evaluator take their fail mode from deployment configuration (`FAIL_OPEN` , shipped as `true` — fail open — in both compose env templates) and do not consult the tenant setting; aligning them is open work. **Use case:** fail-closed for trading-floor endpoints, fail-open for a low-risk pilot group. **Risk:** security tools that fail open silently are a bypass; ones that fail closed unexpectedly are an outage. Either way, it should be your call.

16 · Endpoint / laptop agent coverage — **Working** (macOS/Windows/Linux; macOS kernel sensor Pilot; Windows user-mode ETW sensor Roadmap — no driver, not yet run on Windows, and enforcement does not depend on it) · Teams: SEC

The optional depth layer: process, network, and file monitors; the AI-tool detector (known-tool database + flagging of *unknown* processes talking to AI APIs); MCP-configuration verdicts (row 20); model-file scanning (row 4); DLP at the source; clipboard monitoring; patch inventory/remediation; A-BOM collection. **Deployment reality, stated plainly:** we know no one wants another agent. The platform is built so the agent is *one* deployment option, scoped to the populations where local AI tools actually appear (developer machines), not a prerequisite for the platform. **Risk:** local AI tools — a CLI agent installed without approval — are invisible to every browser and network control at the moment of installation.

17 · Browser surface coverage — **Working** (Chromium/Edge/Brave; Firefox and Safari generated from the same source, not yet validated in those browsers) · Teams: SEC

MV3 extension: pre-navigation URL trust-gate checks, in-browser DLP and redaction, upload control, verdict banners users can understand. Rolls out via MDM with no agent. **Honest scope:** a dedicated browser-security product will carry deeper in-browser features than a seven-surface platform carries in the browser alone. For an organisation that already owns a browser-DLP plugin, this surface is optional rather than the pitch. **Risk:** browser-borne AI usage and data entry — for orgs that don't already have a browser DLP control.

18 · Developer tools / IDE surface — **Pilot** · Teams: ENG, SEC

Extensions for VS Code, Visual Studio, Cursor, and Kiro, plus hooks for Claude Code — local secrets/PII DLP where AI-assisted coding actually happens, with redact-on-save in VS Code, Cursor and Kiro and warn-only in Visual Studio; prompt-submit interception ships as the Claude Code hook. **Honest scope:** tenant policy is synced to VS Code but not yet consulted in-editor — enforcement mode is an editor setting today, and the pattern set is built into each extension. **Use case:** a developer's AI assistant is about to send a file containing credentials as context; the credential is redacted in the editor. **Risk:** source code and secrets flowing to AI services through developer tools that never touch a browser.

19 · AI traffic gateway (network layer) — **Working** (proxy + router); firewall/EDR telemetry fusion in active development · Teams: SEC

Three network-layer options, in increasing depth: (a) a containerized transparent/explicit proxy that intercepts AI traffic and enforces policy inline; (b) an AI provider router — one gateway to 8 providers with a credential vault, request normalization, and cost tracking — which can sit behind *your* existing AI gateway rather than replacing it (your gateway makes one REST call per request for a verdict); (c) **in active development now:** consuming the telemetry your existing next-generation firewall and EDR already produce, correlating it into named findings, and pushing enforcement back through those same tools (dynamic block lists your firewall consumes natively, EDR containment) — AI-specific control with zero new agents and zero traffic re-routing. **Risk:** AI usage that never touches a browser or an approved gateway.

Part 2 · Prove, deploy and trust (20–37)

20 · Shadow-AI / AI-app inventory — **Working** on the agent; agentless path in development · Teams: SEC, RISK

Continuous inventory of AI software and usage: a signature database of known tools (desktop apps, CLIs, IDE plugins, browser extensions), detection of *unknown* processes connecting to AI provider APIs, MCP-server inventory (including plugin-bundled configurations — on a measured developer machine, 16 of 16 real server definitions lived in plugin trees the user-level config never mentioned), and A-BOM correlation. Findings carry severity verdicts and named users, not just an inventory row. **Use case:** the Codex-installed-without-approval scenario — surfaced as “unauthorized AI tool, this user, this host, this destination,” not a row in an asset dump 40 minutes later. **Risk:** ungoverned AI is invisible until it's an incident.

21 · Code scanning + runtime protection — 9 languages — vuln scanning **Working** ; RASP **Pilot** in all nine languages · Teams: ENG, SEC

Two distinct things, often conflated. (a) *Code scanning:* an SCA-style inventory of AI components across your repositories (plus endpoints and cloud), matched against vulnerability data (OSV) and prioritized by CISA KEV known-exploited status and EPSS exploit-prediction scores — so “which vulnerable AI libraries do we actually run, and which are being exploited in the wild.” (b)

Runtime protection (RASP): an in-process layer for your own application servers — Java, .NET, Python, Node.js, Go, Rust, Ruby, PHP, C/C++ — that sits inside the runtime, sees the AI calls and external content your app makes and receives, consults the trust gate before outbound fetches (Python only today), and enforces policy in-app. Think WAF-adjacent, but inside the application, AI-aware. **Use case**: the portfolio-analysis agent's backend gets pre-ingestion checks and policy enforcement without changing app code. **Risk**: AI risk *inside* your applications, where network controls can't see.

22 · Robotics / embodied-AI control surface (ROS 2) — Pilot (validated on real hardware) · Teams: —

A security agent for ROS 2 robotics: topic monitoring, actuator policy enforcement, latching emergency stop, sensor-integrity checks. **Validated on real hardware, and here is precisely what was and was not proven**. Working on the robot: the latching emergency stop held the actuator command at zero on the wire; speed-limit enforcement was verified *on the wire* — a 9.0 m/s command against a 2.0 m/s limit was delivered downstream as 2.0 m/s; and telemetry reached the control plane. Not completed: control-plane **policy sync**, because the bootstrap-token retrieval was unfinished when the test hardware was returned. So the safety layer enforced from local configuration exactly as designed; what remains unproven is remote policy *distribution*, not enforcement. **Enforcement is a deployment choice, and the default is observe**. With `actuator_output_topic` unset the agent evaluates every command and reports violations, but the robot receives exactly what it would have anyway — the clamp and the latching e-stop are recorded, not applied. Interposition, the mode validated on hardware, means setting that topic and rewiring the motor driver to subscribe to it.

Why local-only enforcement is a deployment mode, not a degraded one. Most robots are not internet-connected. A safety agent that enforces limits and holds an emergency stop from local configuration — with no control plane reachable — is the operating mode a disconnected fleet actually needs, and it is the same air-gap property as 33. Remote policy sync is an enhancement for connected fleets, not a prerequisite for the control.

Included for one reason: it exists and it was validated on hardware. Skip it unless embodied AI is in scope.

23 · AI-agent identity, delegation chains, revocation — Working · Teams: SEC, AI

Non-human AI agents become first-class identities: registered with declared capabilities (allowed/denied tools) and a per-agent credential TTL, issued short-lived scoped JWTs, with agent-to-agent delegation chains and token revocation. **Honest scope on attestation**: the workload endpoint issues SPIFFE-format IDs but does not attest — it does not contact a SPIRE server and does not verify the workload it names. **Honest scope on revocation, verified against source, because “instant” needs splitting three ways**: API-key revocation and portal-session revocation work and take effect immediately. **Agent-token revocation is implemented but not yet consulted in-line** — the validation endpoint checks the revocation list and returns `TOKEN_REVOKED` correctly, but no enforcement point calls it today, so revoking an agent token does not by itself stop traffic until the token expires. Short credential TTLs are what bound the window in the meantime. Wiring the enforcement points to the validation endpoint is a known gap, not a design position. **Use case**: your customer-facing agent and every internal automation gets an identity; “which agent did this, under whose delegated authority” has an answer. **Risk**: agents acting with shared, unscoped, unrevocable credentials — the service-account problem, multiplied by autonomy.

24 · Privileged-action broker (kill · isolate · patch) — Pilot · Teams: SEC

Privileged endpoint actions go through one audited, entitlement-gated broker with approval workflows, maintenance windows and per-app auto-approve rules, every action logged before and after execution. Membership of the broker's allow-list *is* the security boundary: an operation not in the table can never execute, whatever a caller sends. **The allow-list is seven operations**: apply patch, isolate host, kill process, quarantine file, and three proxy-configuration operations. **Honest scope, verified against source**: of those six, **apply patch** and **isolate host** are reached by shipped code today; **kill process** is implemented and allow-listed but nothing in the agent currently calls it, and the control-plane command dispatcher handles only patch operations. **Quarantine file** is inside the broker: it carries the before-and-after audit record, and its validator refuses relative paths, symlinks, non-files, anything already quarantined, and a fixed list of protected system prefixes. It is not reachable from the control plane — the command dispatcher handles only patch operations — so it is an endpoint-local action that is now audited, not a new remote one. **Use case**: response to an unauthorized AI tool can be automated *with* governance — approvals and audit, not a script someone runs. **Risk**: response actions that are either unaccountable or so heavyweight nobody uses them.

25 · The block tells the end user who decided, and why — Working (browser surfaces; recent) · Teams: SEC

When something is blocked or redacted, the person sees which policy fired and the category of detection, with an evidence id their SOC can pull — a banner/interstitial with the reason, not a mystery failure. **Use case:** fewer helpdesk tickets, and users learn the policy instead of learning to route around it. **Risk:** silent blocks teach users the tool is broken; explained blocks teach them the rule.

26 · Hash-chained, tamper-evident decision ledger — Working · Teams: RISK, SEC

Decisions from the proxies, the URL trust gate, the AI router and the response service are HMAC-SHA256-signed and hash-chained — each entry cryptographically bound to the previous — with an integrity-verification endpoint that proves any given record hasn't been altered. Plus a directed action graph: agent → model → tool → human lineage for every AI interaction. **Honest note:** the audit chain's signature is HMAC-SHA256 (symmetric), not post-quantum; the PQC suite (row 32) covers key transport, signing, and corpus manifests. **Use case:** nine months from now, prove to an examiner exactly what the control decided on a specific date, and that nobody edited the record. **Risk:** logs that can be silently modified are testimony, not evidence.

27 · Per-decision evidence sealing (every allow / block) — Working · Teams: RISK

Each verdict — including *allows* — is stored with the evidence that produced it: input fingerprint, scores per engine, the policy that matched, redirect chains and IOCs for URLs, decision lineage. **Honest scope:** a full evaluation seals a record. Two URL-gate paths do not write one today — a verdict served from the reputation cache on the fast path, which is what every live caller requests, and a tenant allow/block-list match — so on those the decision is carried by the surface's own audit entry rather than a new evidence record. Sealing them is open work. This is a different thing from logging: a log or an export is a record of activity; this is evidence bound to the control decision that produced it. **Use case:** an examiner asks not “do you have a policy” but “show me this decision” — and you can. **Risk:** compliance credit claimed on controls that can't evidence individual decisions.

28 · Real-time SIEM streaming — 6 destinations — Pilot · Teams: SEC

Native per-tenant forwarding of AI-proxy traffic decisions to Splunk (HEC), Microsoft Sentinel, IBM QRadar, Elastic, Google SecOps, and generic Syslog/CEF — with per-connector configure/test endpoints. Your SOC keeps its single pane; CyberArmor feeds it. **Use case:** every AI-security finding lands in your existing Splunk/Sentinel pipelines and playbooks from day one. **Risk:** another console nobody watches.

29 · Mapped to 45 frameworks / 701 controls, grouped by region — Working (44 of 45 carry installable packs; assessment depth expanding with design partners) · Teams: RISK

One-click policy packs mapped to 45 registered frameworks totaling 701 controls, grouped by the region they apply in so a buyer finds their own regulator rather than scanning an alphabetical list. **Global (9):** ISO 27001, ISO/IEC 42001 (AI management), PCI-DSS, NIST CSF 2.0 / 800-53 / AI RMF, CIS v8, CSA CCM, OWASP (Web+API+LLM+Agentic), SANS Top 25. **North America (9):** SOC 2, HIPAA, SEC Cybersecurity Disclosure, FINRA Cybersecurity, NYDFS 23 NYCRR 500, CMMC L3, CCPA/CPRA, Canada (PIPEDA, Quebec Law 25, OSFI B-13). **Europe (3):** GDPR, EU AI Act, UK (UK GDPR/DPA 2018 with FCA-PRA operational resilience). **Asia-Pacific (17):** Singapore MAS TRM with the binding Cyber Hygiene Notice, Hong Kong HKMA TM-G-1 with C-RAF 2.0, China PBOC/NFRA with MLPS 2.0, Japan, South Korea, Australia, Malaysia, Thailand, Indonesia, Philippines, plus seven Pakistan thematic planning frameworks. **Africa (6):** Nigeria, South Africa, Kenya, Ghana, Egypt, and the AU Malabo Convention as a continental baseline. **Latin America (1):** Brazil LGPD with the BACEN resolutions. Enabling a pack seeds enforceable policies, not a checklist. *(Counts runtime-enumerated from the registry on 2026-10-03: 45 frameworks, 701 controls, 44 policy packs.)*

Honest scope: the national control text paraphrases published requirements for self-assessment and is not a reproduction of any statute or licensed standard; registration thresholds and reporting deadlines need local counsel before they go in a contract. One framework, SANS/CWE Top 25, deliberately has no pack, because its controls are software weakness classes that an AI-egress policy cannot evidence. **Pakistan:** the National AI Policy, PISF, cyber/CERT, cloud/procurement, SBP, privacy preparation and deployment-readiness mappings preserve the workbook obligations. Three configuration checks verify the framework's enabled redaction, monitoring and provider-restriction policies; a fourth checks recorded tenant AI activity. Governance requirements need customer documents and show pass · attested when fulfilled. Configuration and attestation remain distinct from observed enforcement and from certification. **Risk:** AI governance that exists in a policy PDF but not in any enforcement path.

30 · Examiner-ready compliance exports — Configurable · Teams: RISK

Per-framework assessment reports — scored, stored, tenant-scoped, with evidence bound to each control decision — exportable for an examiner, auditor, or internal risk review. **Use case:** FINRA exam prep goes from a quarter of screenshot archaeology to an

export. **Risk:** controls that work but can't be demonstrated cost nearly as much as controls that don't work.

31 · EU AI Act evidence alignment (Art. 99 regime) — `Working` · Teams: RISK

A dedicated EU AI Act framework now ships: 17 controls covering risk-tier classification and prohibited-practice screening first, because nothing else is assessable until a system's tier and the organisation's role are known, then the high-risk duties — risk management, data governance, technical documentation, Art.12 automatic record-keeping, transparency, human oversight, and Art.15 robustness against manipulation including prompt injection — plus GPAI obligations, post-market monitoring and serious-incident reporting. Its policy pack enforces three of those directly: prompt-injection warning against Art.15, AI-interaction recording against Art.12, and redaction against Art.10 data governance. **Honest scope:** conformity assessment, CE marking, EU-database registration and the fundamental-rights impact assessment are organisational acts, not technical controls, and are marked for manual review rather than scored as passes. The ISO/IEC 42001 and NIST AI RMF packs plus the per-decision evidence ledger (rows 26–27) remain the operational backbone an Art. 99 conversation turns on. This matters to any organisation shipping EU-facing AI features.

32 · Post-quantum crypto executing — ML-DSA-87 + ML-KEM-1024 — `Working` · Teams: SEC

NIST-standardized post-quantum algorithms running in shipped code paths — ML-KEM-1024 key encapsulation for API-key transport (AES-256-GCM payloads), ML-DSA-87 signatures (with Ed25519 fallback where liboqs is unavailable), signed detection-corpus manifests with anti-rollback and expiry. **Honest scope:** the liboqs runtime is compiled into every service image and the ML-KEM-1024 / ML-DSA-87 primitives are called natively, but PQC API-key transport is OFF in the default and shipped-production config (`CYBERARMOR_PQC_AUTH_ENABLED=false` , plaintext keys still accepted) and is enabled per deployment. Where liboqs is absent the KEM falls back to classical X25519. **Use case:** harvest-now-decrypt-later is a real concern for a financial firm's long-lived records; CNSA 2.0 timelines are already published. **Risk:** re-platforming cryptography later, under deadline.

33 · Fully offline / air-gapped deployment — `Configurable` (delivery mechanism built 2026-09-07; four items still gate a first install) · Teams: SEC

Customer data never leaves the deployment, and that part is unqualified. Detection runs against locally loaded models with `TRANSFORMERS_OFFLINE` enforced, and the seeding step *verifies* each model cold-loads with the offline flags forced on rather than assuming it. OCR weights ship inside the wheel. The known-good hash corpus verifies signed manifests with keys pinned in the agent config that are never fetched, and its artifact URLs live inside the signature, so they can already point at a customer-run internal host. The macOS `.pkg` and the Windows bundle carry their own CPython and a full wheelhouse and install with no network at all.

Delivery to an isolated network is built but not yet exercised at full size. `scripts/deployment/build_offline_bundle.sh` saves every image compose refers to (35 across both profiles) with a manifest; `install_airgap.sh` loads rather than builds and refuses a tampered, partial or secret-less bundle. A *full* bundle has never been built end to end and no install has run on a genuinely isolated host, which is why this row is `Configurable` and not `Working`.

Four items still gate a first install, and the row will not move until they are closed: the model-seeding script cannot load a model from a local path, the post-quantum base image clones liboqs from GitHub during the build, the Linux endpoint installer still pip-installs from the public index, and the URL crawler's SSRF guard rejects private ranges by default. Two capabilities additionally have no offline mode and fail *quietly* rather than loudly: A-BOM vulnerability scanning (OSV/KEV/EPSS are network calls) and Google Safe Browsing (it keeps no on-disk state). Mirror those feeds internally or record them as unavailable in that deployment. See `docs/runbooks/airgapped-install.md`. **Use case:** the AI security layer inspects your most sensitive traffic without your data ever leaving your network — including to us. **Risk:** an AI-security vendor that is itself a data-exfiltration path.

34 · Self-hosted + customer-cloud + hosted options — `Working` · Teams: SEC

Two shapes are complete today: your data center on Docker Compose, or hosted by us — and on those, no capability is hostage to one model. **Honest scope:** the Kubernetes Helm chart covers a minority of the platform's services and is not current; there is no Terraform module. Self-hosted deployment requirements are documented separately. **Use case:** a large institution self-hosts everything; a smaller advisory firm takes hosted; an MSSP runs it multi-tenant. **Risk:** "cloud-only" is a non-starter in exactly the verticals that need this most.

35 · Multi-tenant MSSP wholesale architecture — `Working` (console shipped; partner program forming) · Teams: —
A dedicated MSSP console for partners managing subsets of tenants, with tenant scoping enforced on the customer-portal session plane and on every tenant-scoped agent, policy, telemetry and artifact API route — each API key is pinned to the tenant it was issued for, and a key that names another tenant, in the path or in a request body, is refused. **Honest scope:** the `x-role` and `x-tenant-id` request headers are still honoured ahead of the key's own role on the admin API plane; removing that precedence is open work. Relevant to a direct enterprise buyer mainly as an architecture proof: strict multi-tenancy is a security property, not just a business model. Relevant to an investor as the wholesale distribution path.

36 · Enterprise SSO / MFA / multi-tenancy — `Working` · Teams: SEC
OIDC SSO with PKCE and just-in-time provisioning against Entra ID, Okta, Ping, and AWS IAM Identity Center; TOTP MFA (RFC 6238) with backup codes, switched on per tenant and enrolled per user — **enabling it for a tenant does not yet block a user who has not enrolled:** they still sign in with the email code alone, recorded as a login-without-MFA event rather than denied (an enrolment gate is open work); and directory-identity enrichment — findings and audit events resolve to a named user, department, and group, which is what makes row 20's findings actionable. **Use case:** “unauthorized AI tool” arrives as a person and a department, not an IP address.

37 · Independent & vendor-neutral — structurally — a fact, not a feature · Teams: —
CyberArmor is independently held and owns no adjacent platform, which is what makes the bring-your-own-stack architecture credible: we consume your firewall's telemetry, feed your SIEM, sit behind your AI gateway and coexist with your EDR because we compete with none of them. For a buyer, vendor neutrality means the product's incentive is your existing stack working together rather than being displaced.

Statuses reconciled against the CyberArmor source repository on 2026-08-19. Where a capability has a known limit it is stated in its entry rather than omitted. Questions on any capability — including “show me” — are welcome; most have a runnable demonstration.

Part 3 · Capabilities folded inside the entries above

These ten capabilities are in the product and evidenced in the source, but were folded inside broader entries above where a reader scanning titles would not find them. They are numbered separately so nothing is double-counted.

38 · Honest coverage reporting — a failed check never reads as clean — `Working` · Teams: SEC, RISK
Scan responses carry `scan_complete` and `detectors_unavailable`, so a detector that could not run surfaces as an explicit finding with `assessed: false` rather than as silence. `GET /ready` reports true per-model state — loaded, failed, unavailable, or not attempted — and returns *degraded* rather than asserting health it has not verified. Endpoint monitors follow the same rule: an agent that cannot read a watched directory reports degraded and names the path. **This is distinct from 15.** Capability 15 decides what *happens* when a component is unreachable; this decides whether the *result* admits its coverage was incomplete. A tool can fail closed and still hand you a clean scan it never actually ran. **Use case:** your dashboard is green — this tells you whether that means “we checked and found nothing” or “we could not check.” **Risk:** the most expensive failure in security tooling is not a missed detection, it is a missed detection reported as a pass.

39 · Ransomware and mass-deletion: detection on by default, host containment on request — `Pilot` · Teams: SEC
Three behavioural signals over user-data directories: **mass deletion** (100 files in a 60-second window), **mass encryption by rename** (5 files renamed to known ransomware suffixes within 60 seconds), and **in-place encryption** (Shannon entropy ≥ 7.5 bits/byte over the first 4 KB of rewritten text, source, config and log files). Scope is each user profile's Documents, Downloads and Desktop plus a temp directory. This is destructive-*phase* detection — it catches the behaviour, not the binary, so it does not depend on recognising a family. **Honest scope:** a behavioural signal on user-data paths, not a full anti-ransomware product, and not a replacement for your EDR.

What is on out of the box: the three signals, as telemetry. A full ransomware run produces alerts and no automatic action.

How protection is switched on, and what it does. One action exists — the endpoint cuts its own outbound network access, permitting only the control plane so the machine stays manageable (capability 44). It is **off by default at three independent points**, each of which must be satisfied: a tenant master switch (default off), a per-detection action that defaults to *report only* rather than *isolate*, and the agent's own local default, which stays off until it has successfully fetched the setting and reverts to off on any unreadable response. A `tenant_admin` arms it in the customer portal under **Settings** → **Automated endpoint response** by enabling the master switch *and* changing each detection's dropdown from "Report only" to "Isolate the endpoint" — both are required, and arming only one does nothing. There is no environment variable or endpoint-side flag: the setting is server-issued only. Endpoints collect it on their next signature-intel poll, so **arming can take up to an hour to reach the fleet**; there is no push.

What it does not do, stated because these are the things people assume: no file quarantine from these triggers, no rollback or snapshot of already-encrypted files, and none of these events reach the Incidents view — they are telemetry and alerts. **Process kill is now reachable from these triggers** (capability 66) and **recovery-path tampering is now detected** (capability 65); both were absent before 2026-08-28 and are described in their own entries. Containment is verified by hand on Linux, and **on macOS it is now verified in pf itself** — the anchor is attached to the main ruleset and the loaded state is read back before success is reported (capability 44). The Windows path has never been executed end to end, so treat it as unproven. **Recovery matters as much as containment:** an isolated Linux host drops sshd's reply packets too, so recovery needs console or physical access, and there is no console "release" command yet. **Use case:** an AI coding assistant is induced to run a destructive script; the file behaviour is flagged as it happens. **Risk:** capability 10 detects ransomware *shape in generated output*; this detects the destructive phase actually executing on disk.

40 · Vulnerability scanning with exploit-based prioritisation — `Working` · Teams: ENG, SEC

CVE matching over the AI Bill of Materials via OSV, enriched and prioritised by **CISA KEV** known-exploited status and **FIRST EPSS** exploit-prediction scores — so the question answered is not "how many CVEs do we have" but "which vulnerable AI components do we actually run, and which are being exploited in the wild right now." Inventory spans endpoints, source repositories and cloud sources. **Note:** this sits inside row 21, whose title emphasises runtime protection; it is called out separately here because it is `Working` today and is the row an application-security team will ask about first. **Risk:** vulnerable AI/ML dependencies prioritised by CVSS alone waste the remediation budget on things nobody is exploiting.

41 · AI Bill of Materials (A-BOM) — `Working` · Teams: RISK, ENG

Component inventory of AI artifacts — models, datasets, libraries — across endpoints, source repositories and cloud sources, with provenance. It is the substrate 5, 20 and 40 all read from: the vulnerability scan matches CVEs against it, shadow-AI findings correlate to it, and supply-chain questions resolve against it. Called out separately because "*what AI artifacts exist in this organisation and where did each come from*" is a question examiners and internal audit ask directly, and it deserves an answer that is not buried inside three other rows. **Risk:** you cannot govern, patch or attest to AI components you have never inventoried.

42 · Patch remediation through the audited broker — `Pilot` · Teams: SEC, ENG

Endpoint patch remediation across **winget, Homebrew, apt and yum/dnf**, entitlement-gated, with maintenance windows, an approval workflow and per-app auto-approve rules — executed through the privileged-action broker (24) so every action is logged before and after. **Use case:** a vulnerable AI library flagged by 40 is remediated on the affected endpoints with approvals and an audit trail, rather than by a script someone runs by hand. **Risk:** knowing which component is exploitable and having no governed path to fix it.

43 · Automatic quarantine of a malicious executable — `Working` · Teams: SEC

When an executable appearing in a watched user directory is judged critical, the agent moves it to a quarantine directory and strips its permissions (`chmod 000`). **Exactly three things reach that verdict:** a malware signature matched by the detection engine, a known-bad file hash, or a **high-signal content pattern together with independent corroboration** — encoded PowerShell, fileless-injection API calls or a fetcher piped into an interpreter, *and* the file arrived from the internet or sits in a temp directory. **On Windows the move is the containment** — Python's `chmod` there only toggles the read-only bit and cannot make a file inaccessible. **This is on by default and no human approves it** — it is the one destructive response the product takes without being switched on first, so it is called out rather than left for a customer to discover. Weaker indicators on their own — a temp path, `sh -c`, a shell invocation — raise the finding to medium and are reported, never acted on. A high-signal pattern with no

corroboration is flagged, not moved. Being unsigned is deliberately *not* corroboration: shell scripts are never signed, so treating it as evidence would make it a constant rather than a signal. Everything else is surfaced as a flagged telemetry event. **Honest scope:** this runs outside the privileged-action broker (capability 24), so it does not carry the broker's approval workflow or its before-and-after audit record; bringing it inside is open work. **Use case:** a dropper written to Downloads by a compromised AI tool is neutralised before it is run. **Risk:** signature-matched malware sitting executable on disk while a response waits for someone to approve it.

44 · Automated host containment on destructive file behaviour — Configurable · Teams: SEC

The one automated response to the capability-39 signals: the endpoint cuts its own outbound network access, permitting only the control plane so the machine can still be managed and released. IPv4 and IPv6 are both cut. Configurable rather than Working is deliberate: it is off by default at three independent points (capability 39 names them and how to arm it), so a tenant that has not deliberately armed it gets telemetry and no action. **Honest scope:** verified by hand on Linux, and **macOS is now genuinely enforced and verified** — as of 2026-08-28 the pf anchor is attached to the main ruleset and three facts are read back out of pf (pf enabled, anchor listed, block rule present) before success is reported, so a host that cannot be contained now says so instead of reporting success. Windows has never been executed end-to-end. Isolation is OUTPUT-only, so containers, VMs and WSL2 on the host are not covered, and DNS is permitted broadly rather than only to the control plane's resolver. The action runs through the privileged-action broker, but this call site does not pass the central audit sink, so the record lands in the endpoint's local log rather than the tamper-evident trail (capability 26) — closing that gap is open work. **Use case:** a machine that has begun mass-encrypting files stops being able to reach anything but the console, without waiting for an analyst. **Risk:** the window between detecting a destructive process and a human reacting to it.

45 · Citation and link trust-gating inside model answers — Working on proxy surfaces · Teams: SEC, AI

URLs a model *returns* are evaluated before the user can act on them, not merely the URLs a user sends. The trust gate resolves redirect chains, checks reputation and structural signals, and the surface enforces the verdict by withholding the whole model response behind a block page — there is no per-link rewrite or interstitial, and redact verdicts are not enforced on this path. **Honest scope:** response-side evaluation ships on the two MITM proxy surfaces; the extension, SDK and RASP surfaces carry request context only. **Withholding is what a streamed answer cannot do.** Since 2026-09-20 a text/event-stream response is forwarded frame by frame (see capability 10 for why), so there is no whole response left to withhold and the trust gate is not consulted on that path at all; the flow is recorded with a coverage gap that says so by name. **Use case:** a model confidently cites a source that is a credential-harvesting page, and the link never becomes clickable. **Risk:** treating model output as trusted content because it came from your own approved assistant.

46 · SaaS OAuth consent revocation — Working · Teams: SEC, RISK

Third-party OAuth grants into connected SaaS tenants are inventoried and can be revoked through the integration-control API — Microsoft 365 consent grants via Graph, Google and Salesforce by token revocation; there is no console UI for it yet, closing the standing access an approved-then-forgotten AI integration keeps. **Use case:** a productivity add-in with read-all-mail scope is revoked grant by grant. **Risk:** OAuth grants are the access path least likely to be reviewed and the most likely to outlive the tool that requested it.

47 · The interactive allow / deny gate — Working (macOS endpoint proxy) · Teams: SEC

Where policy is set to prompt rather than decide, the endpoint local proxy holds the request, shows the user what was detected and which policy fired, and waits for a live allow or deny — recorded, and distinguishing “the user said no” from “we could not ask”. **Honest scope:** the held prompt is macOS-only today. **Use case:** a genuine business need to paste a client reference into an AI tool is not blocked outright, but it is deliberate, attributable and logged. **Risk:** a control that can only block or allow gets configured to allow.

The three questions this document is built to answer

For a CISO. *What does it do, what does it not do, and can I prove it to an examiner?* Every row carries a status, and every known limit is written into the row rather than left to discovery — see 4, 5, 10, 11, 17, 26 and 31, each of which states a boundary. Capabilities 26, 27, 29 and 30 are the evidence chain.

For a risk or compliance leader. *Does the control exist in an enforcement path, or only in a policy document?* Enabling a framework pack seeds enforceable policies, not a checklist (29), and every verdict — including allows — is sealed with the evidence that produced it (27).

For an investor. *What is actually built, and how far past a demo is it?* Every capability carries a maturity status reconciled against the source repository, and the boundaries are written in rather than omitted. The evidence, deployment and agent-identity entries (26, 27, 29, 30, 23, 24) are where the platform extends past prompt filtering.

Part 4 · Agentless — work through the security stack the customer already runs (48–52)

These five are new and they answer one objection directly: “*nobody wants another agent.*” Instead of installing a sensor, CyberArmor consumes the telemetry the customer’s firewall and EDR already produce, correlates it, and pushes enforcement back out through those same controls. Every entry here is `Pilot`: the code runs and is tested end to end against recorded vendor formats and a simulated feed, and **none of it has yet been exercised against a live Palo Alto instance**. That sentence is the status, and it is here rather than in a footnote.

48 · Shadow-AI detection with nothing installed on the endpoint — `Pilot` · Teams: SEC, AI

The customer’s existing sensors do the seeing. CyberArmor ingests firewall logs (PAN-OS over syslog, in the vendor’s CEF format or the native comma-delimited one), EDR process and network telemetry, any CEF sender, or a push from the SOC’s own aggregator — normalises all of it into the one event taxonomy, and tags it against the shared catalogue of AI services, domains and tool process signatures that the endpoint agent uses. The result is that an unapproved AI tool is discovered on a machine that carries no CyberArmor software at all. **Volume is handled at the source**, not by drinking the firehose: the onboarding guide configures the firewall’s log-forwarding profile to AI-relevant categories, and the listener additionally rate-limits, de-duplicates and bounds what arrives anyway — every one of those bounds sheds with a counter rather than silently. **Ingestion fails open by design** — a dead feed never blocks anything — so the failure is made loud instead: a silent source raises a `telemetry_feed_stale` event, a warning, and visible lag. **Use case:** an engineer installs a CLI coding agent; the firewall classifies the outbound call and the EDR sees the process, and the finding appears without anyone touching the laptop. **Honest scope on the EDR half:** the only EDR adapter built today is Wazuh, used as a lab stand-in; Defender, Falcon and SentinelOne connectors are not built. Any other EDR feeds through the generic CEF or webhook path, or from the SOC’s own aggregator. **Risk:** the tools that already see shadow AI mostly only label it, and by the time anyone reads the report the session is over.

49 · Correlated findings with a person’s name on them — `Pilot` · Teams: SEC, RISK

Two sightings become one card. A process event from the EDR and traffic to a catalogued AI destination from the firewall, on the same host inside a two-minute window, produce a single `unauthorized_ai_tool_detected` finding carrying the tool, the user, the host, the destination and **every sensor that saw it** — not two alerts in two consoles. A network sighting with no matching process opens a zero-day finding that **upgrades in place** when the process half arrives, because two cards for one incident is how an analyst double-counts. The rules are deterministic and explainable on purpose: an evaluator who runs the firewall daily will ask *why* two things correlated, and “they share a host inside a window” is an answer where a score is not. **Severity is honest about what it does not know** — while the tenant’s approved-provider list has not been read, a finding caps at *high* rather than claiming *critical*. **Use case:** the console says “Codex CLI, unapproved, jsmith in Equity Research, WKS-1234, seen by your EDR and your firewall.” **Risk:** an IP address is not an employee, and a finding nobody can attribute is a finding nobody actions.

50 · Firewall-enforced AI allow list (External Dynamic Lists) — Pilot · Teams: SEC

CyberArmor publishes the tenant's AI policy as plain-text External Dynamic Lists that a PAN-OS firewall consumes natively over HTTPS: the catalogue of AI services **minus** the providers that tenant has approved, plus incident-driven blocks that each carry an expiry so a temporary block cannot quietly become permanent policy. Enforcement happens on the customer's own firewall, list updates need **no firewall commit**, and turning it off is deleting one rule. **The honest limit, because it is the one a good evaluator will find:** this is a default-deny bounded by a catalogue we maintain, so a genuinely novel AI destination nobody has enumerated is not blocked by it. Pair it with the firewall's own AI URL category — the vendor maintains that list — and the two together are a real allow list; ours alone is not. **Failure behaviour is decided rather than incidental:** a policy-service outage serves the last known-good list with its staleness measured and exposed, and a tenant whose approvals have never been read gets an explicit refusal rather than a guessed list, because guessing full-catalogue would block providers they approved and guessing empty would block nothing while claiming coverage. **Use case:** unapproved AI services are blocked at the perimeter, and approving one is a policy change that takes effect at the next refresh. **Risk:** discovery without a control is a report, and reports do not stop data leaving.

51 · Enforcement round-trip to the customer's firewall — Pilot · Teams: SEC

When a finding warrants it, CyberArmor pushes a real action back into the customer's stack: a commit-less tag on the source host through the firewall's User-ID API, matched by a dynamic address group the customer configured in advance, and destination blocking through the list in 50. **Every action can be dry-run first** — the preview shows the exact request that would be sent and sends nothing — **every action names its own rollback**, and a playbook returns the reverse-ordered undo for the whole run. Each action is written to the signed audit trail before and after it executes, and the credential is fetched from the secrets service at call time rather than stored in the service. **Deliberately excluded:** anything that mutates a security rule or requires a commit. Banks will not permit it and it is not reversible in one call, so it is documented as pre-agreed configuration instead. **Use case:** the host is quarantined and the destination blocked while the analyst is still reading the finding. **Risk:** the gap between detecting unsanctioned AI use and doing something about it, measured in the time it takes a human to open a second console.

52 · Authenticated sensor feeds (mutual TLS syslog) — Pilot · Teams: SEC

Sensor feeds can be authenticated with a client certificate instead of identified by their network address. This matters more than it sounds: the ordinary way syslog is received — trusting the sender's source address — **is not authentication**. A UDP source address can be forged, and behind NAT an address identifies a whole network rather than one device, so every host behind the customer's egress address is implicitly trusted to write into their tenant. On the mutual-TLS listener (RFC 5425) the sender proves possession of a key issued by the customer's own certificate authority, the address is never consulted, and a feed keeps its identity when it legitimately moves. A certificate identity **retires** the address route for that source, so a feed cannot be described as authenticated while still accepting unauthenticated traffic; the listener refuses to start on a partial configuration rather than coming up looking secure; and each configured feed reports which of the two modes it is actually using. **For senders that cannot present a client certificate**, the guidance is to carry plain syslog inside a VPN or IPsec tunnel and restrict the port to it — not to expose it and hope. **Use case:** the firewall's log feed is cryptographically identified, so fabricated events cannot be injected into a tenant by anyone who learns its egress address. **Risk:** telemetry an attacker can forge becomes findings naming real employees, and an analyst acts on them.

Part 5 · Built and in use, documented here for the first time (53–60)

These shipped without a capability entry. They are listed now because each is user-visible and each answers a question a buyer actually asks — and because a capability nobody wrote down is one the customer never learns they bought.

53 · Document and image content inspection with offline OCR — Working · Teams: SEC, RISK

The reading half of capability 11, called out because it is the part buyers assume is missing. PDF, DOCX, XLSX and PPTX are extracted at rest, and images are read with **OCR that runs locally** — no cloud vision service, so it still works in an air-gapped deployment (capability 33). Format is sniffed from the file's magic bytes, never from its name or its declared content type, so

renaming `payroll.xlsx` to `notes.txt` does not get it past inspection. The rule the extractor is built around is that it never reports text it did not actually get: a parser that fails says so, and the finding carries which parser ran, whether extraction completed, and whether OCR was used. **Honest scope:** OCR is accurate on clean screenshots and document scans, not on photographs of screens or on handwriting. **Use case:** someone pastes a screenshot of a customer account page into a chatbot; the account number is read out of the pixels and redacted. **Risk:** DLP that only reads plain text is bypassed by the two most common ways people actually move data — attach a document, or screenshot it.

54 · Endpoint-installed AI-traffic proxy for native desktop apps — `Pilot` (operator-installed; deployment and coverage validation required) · Teams: SEC

The browser extension covers the browser. This covers everything else: the ChatGPT and Claude desktop clients, CLI coding agents, and any native application that talks to an AI service. A local proxy runs on the endpoint itself and inspects that traffic with the same engine as the network proxy — the two are held identical by a parity test rather than by intention. Installation trusts the interception CA in the system keychain as well as the user's, because Chrome does not accept the user one. **It survives a captive portal**, which sounds minor and is not: a proxy that breaks hotel and airport Wi-Fi gets uninstalled by the user in the first week. A direct probe detects the portal, the system proxy is stood down through the audited privileged-action broker so the sign-in page works, and it is restored afterwards. **Honest scope:** macOS and Linux have run. A Windows installer pair (`install_local_proxy.ps1` and its uninstaller) was written on 2026-09-07 using the same shape — WinINET system proxy, per-user CA trust, a logon task, a readiness wait before any traffic is pointed, and an opt-in `-Enforce` that firewalls known AI apps' direct egress — and has not yet been run on a Windows host. **Use case:** an executive uses the Claude desktop app on a laptop in an airport lounge; policy still applies, and the Wi-Fi still works. **Risk:** native AI apps are the coverage gap every browser-only control has, and they are what the heaviest users install first.

55 · The endpoint reports when it is *not* enforcing — `Pilot` · Teams: SEC, RISK

Coverage is asserted from three live authorities that must all agree — the process is running, the port is accepting a connection, and the operating system's network configuration actually points at it, per network service. **File presence is deliberately refused as a signal.** That is not a hypothetical: the installer once answered “is the local proxy installed?” by checking whether the uninstall script existed, and reported “Local proxy detected” on a machine doing zero enforcement, during a live demo on 2026-08-05. A newly plugged-in network interface counts as a coverage gap until it is proven covered. **A fourth authority was added on 2026-09-20: can it enforce the responses it is actually seeing?** Reaching the backends is not enforcing either, and a streamed response cannot be blocked at all (capability 10), so the report now carries how many responses the proxy inspected, how many of those were streamed, and how many of the streamed ones were scanned, truncated, cut off mid-stream or carried a verdict nothing could apply. An endpoint where streaming dominates reads `PARTIAL`, with those numbers, rather than `ACTIVE`. **Use case:** the console can distinguish “this endpoint is protected” from “this endpoint has our software on it”, which are different facts. **Risk:** every deployment metric that counts installs rather than enforcement overstates coverage, and does so most on the machines someone has quietly worked around.

56 · Upload-endpoint discovery, and one click to close the gap — `Working` · Teams: SEC

An AI vendor changes the URL its file uploads post to, and every upload control keyed to the old path silently stops covering it. The extension notices uploads going to endpoints the catalogue does not cover and reports them; the console aggregates thirty days of those sightings by host and path — collapsing per-conversation identifiers so one real pattern does not appear as ten thousand — and offers the administrator a single action to bring each one into coverage. **Use case:** ChatGPT ships a new attachment endpoint on a Tuesday; it appears in Upload Discovery on Wednesday and is covered by Thursday, rather than being noticed after an incident. **Risk:** coverage decays silently. This is the answer to the strongest objection to any catalogue-based control — that the catalogue is always out of date.

57 · Manual control attestation, with provenance carried into the score — `Working` · Teams: RISK

Not every control is machine-observable; some are procedures a human performs. Those can be attested in the console by a named person, with a note and a supporting document. What makes this more than a checkbox is that the compliance score **records how it knows** each control passed, and ranks it: platform-observed outranks customer-attested, which outranks caller-asserted, and absent or indeterminate do not satisfy a control at all. Historical rows that predate the labelling are not quietly promoted to observed — they stop contributing until a fresh assessment re-derives them. **Use case:** an examiner asks which of your passing controls the platform actually verified and which your staff asserted; the report already distinguishes them. **Risk:** a compliance score that blends observation and assertion into one number is the number that fails an audit.

58 · Endpoint protection status the user can see — **Pilot** (macOS menu bar shipped; Windows coverage verification remains) · Teams: SEC

A status indicator showing whether protection is running, from a document the agent writes. **There is deliberately no pause, disable, or uninstall in it** — this is an EDR on regulated endpoints, and a user-reachable off switch is an audit finding. The only exit hides the icon until next login and never stops the protection. The indicator cannot fail green: staleness is judged on the worst of two independent clocks, and a status document that is missing, stale, unparseable or untrustworthy renders as Stopped rather than as unknown. **Validation update, 2026-10-03:** Windows 1.0.6 beta status documents and proxy counters were read on a Parallels VM, replacing the old claim that no Windows host had been checked. The status reported Protected while proxy-installation reporting disagreed with observed traffic; Windows coverage reporting remains pilot work. **Use case:** a helpdesk can ask “what does the icon say” and get an answer that means something. **Risk:** silent agents generate support tickets and quiet uninstalls, and a status light that is green when it should not be is worse than no status light.

59 · Framework and provider integrations for AI application teams — **Working** (Python); other languages vary · Teams: AI, ENG

Drop-in integrations so a team building on a framework gets policy enforcement and an audit trail without writing an integration: LangChain and LlamaIndex callback handlers that enforce on every model call, Vercel AI, Semantic Kernel, and trust-gating for OpenAI and Anthropic tool-use that extracts URLs out of tool-call arguments and gates them before the tool runs. **Honest scope:** Python is the most complete; Node, Go, .NET and Java cover a subset of the same frameworks. **Use case:** the AI team ships an agent that browses; a URL the model produced is checked before the tool fetches it, without that team building a control. **Risk:** governance that only exists at the network edge is invisible to the team actually building the agent, and is the first thing routed around.

60 · Operational reporting over your own data — **Working** · Teams: SEC, RISK

Five reports generated over a chosen window from the tenant’s own telemetry, audit records, incidents, policies, agents and providers: executive summary, AI risk, DLP activity, endpoint health, and policy effectiveness. Distinct from capability 30, which produces per-framework compliance evidence for an examiner — this is management reporting for the people running the programme. The policy-effectiveness report is the useful one and the uncomfortable one: it names policies that have never fired. **Use case:** the quarterly AI-risk review is generated from live data instead of assembled by hand from screenshots. **Risk:** controls nobody reviews drift into either noise or irrelevance, and unused policies are how a rulebook becomes theatre.

Part 6 · Media authenticity and payment verification (61–64)

The deepfake problem, addressed where it can be addressed honestly. A finance team does not need to know whether a voice on a call was cloned; it needs a procedure that does not depend on recognising one. Capability 61 is that procedure. 62 and 63 are the two questions about a file that can be answered from cryptography and metadata rather than from a model’s opinion.

61 · Out-of-band verification for payment instructions — **Pilot** · Teams: SEC, RISK

A payment above a threshold the firm sets is held until somebody phones the counterparty back **on a number from the firm’s own directory of record**, and two named people who are not the initiator confirm it. The directory is the control: a deepfake-enabled wire-fraud call works by supplying a plausible callback number alongside the instruction, so verifying against a number the requester gave you verifies nothing. There is deliberately no code path anywhere in the product that turns a caller-supplied number into a callback target — a counterparty who is not in the directory produces a held payment and an explicit refusal to place the call, which is a different operational state from a payment awaiting one. Approvers are emailed when a payment is held, the queue records **whether that notification actually went out**, and every state change is written to the signed audit chain. Completing a callback satisfies FINRA-FRAUD-1 on platform-observed evidence, not on a customer’s assertion — a tenant that asserts the control passed is overridden by the platform’s own reading of its records. **Honest scope, and it is the whole of it.** The control is built and proven end to end — a payment above the threshold is held, the initiator is refused their own callback, one authorizer is

refused, two close it, and a resolution poll flips from hold to release. What it does NOT do is observe payments by itself. **The platform does not watch your payment rails.** The system that holds the payment posts an event to `POST /payments/events`, and until a firm builds that one integration this control sees nothing and produces no record — a payment nobody sends looks exactly like a payment that cleared, which is why this says `Pilot` and not `Working`. The endpoint is not exposed on a public route today either; standing it up for a firm is part of the same first integration. CyberArmor also holds no funds: recording an outcome produces the evidence, releasing or cancelling happens in the system that holds the money, and the resolution endpoint lets that system poll for the answer. Changing a number of record is restricted to a tenant administrator and attributed. **Use case:** a CFO receives a convincing call and an email chain authorising a \$2m transfer to a new account; the wire is held, the callback goes to the number in the vendor master rather than the one in the email, and the fraud fails at the phone call. **Risk:** this is the control that works whether or not the voice was synthetic — the FBI puts business email compromise losses in the billions annually, and no detector needs to be right for this to stop one.

62 · Content Credentials (C2PA) verification — `Working` · Teams: SEC, RISK

Cryptographic provenance, checked properly, **on audio as well as images and video.** Where a file carries a C2PA manifest, the platform verifies the manifest against the bytes and reports one of five distinct states: no manifest, a manifest that is present but cannot be read, a manifest whose signature is intact but whose signer is not on the deployment's trust list, a manifest that verifies against a trusted signer, or **a manifest that does not match the file it is attached to.** That last one is the only hard positive in the media layer, and it is not a probability — it means the file was altered after signing, or the manifest was copied from another file. The three signature-bearing states are kept apart deliberately: the reference library reports a file signed by an attacker's own certificate authority as "Valid", so a product that keyed off that would certify anything carrying any signature. An unreadable manifest is likewise never folded into "no manifest" — "we could not check this" and "there was nothing to check" are different answers, and only one of them is reassuring. Where a manifest does verify, the platform prints the origin the file declares **verbatim** rather than rendering a tick — the most common verified result in 2026 is a valid, trusted signature stating that a model produced the content, and a green check mark on that inverts its meaning. **Honest scope:** most files in circulation carry no Content Credentials at all, and their absence is not evidence of anything. **Use case:** a board pack arrives with an image that carries provenance; the signature does not match the pixels, and the discrepancy is on the record before the meeting. **Risk:** provenance is the only media signal that is cryptographic rather than statistical, and it is the one an examiner can check independently.

63 · Provenance claim consistency — `Working` · Teams: SEC, RISK

Somebody says "this is the original, straight off my phone." The container says FFmpeg wrote it. Cameras do not write FFmpeg. **This runs on a recorded call the same way it runs on a video** — an audio container names its encoder too. That is a deterministic contradiction between a human's description of a file and the file's own metadata, and catching it needs no model, no training data and no probability. The check runs only when a claim is actually made, and the vocabulary of claims it can check is closed — a claim outside it is refused rather than answered, because replying "nothing inconsistent found" to a question nobody asked would read as endorsement. **Honest scope: there is no "claim verified" outcome and there will not be one.** Metadata can be stripped — every messaging platform does it on upload — and it can be forged, and a deepfake played on a monitor and filmed with a real camera produces a completely genuine camera original. The strongest available answer is that the platform looked and found no contradiction, and it says exactly that. A contradiction is reported as a fact about the file's history, never as an accusation about the person who sent it. **Use case:** a video arrives described as untouched footage from a site visit; it names an editing suite in its own metadata, and the discrepancy is raised before the video is relied on. **Risk:** the cheapest lie to tell about a file is a lie about where it came from, and it is usually told in the covering message rather than in the file.

64 · Synthetic-speech advisory signal — `Roadmap` · Teams: SEC

A model that reports indicators of synthesised speech in audio the platform already receives, as an **advisory band beside a statement of what the model has not been trained on** — never a verdict, never a score reaching a risk decision, and never able to block anything on its own. It is being built rather than licensed: every published detector we audited fails its licence chain at some layer, most commonly in training data restricted to non-commercial research, so ours is a classification head trained on a permissively licensed backbone over data we hold commercial rights to end to end. **Honest scope:** the model is not built and emits nothing. What IS live in production is the surrounding machinery — audio routes into the media pipeline, the two deterministic checks above run on it, and the absent model reports itself unavailable with `assessed: false` rather than returning a clean-looking band. That is the honest-failure plumbing working as designed, and it is not detection: no indicator of synthesised speech is produced today by any code path. The audio model ships first, on the clean licence chain in ADR-0003. video synthesis

classification is planned and not built. It carries a harder constraint than audio and the constraint shapes the design rather than being a caveat on it: published detectors of that kind transfer poorly to generators they were not trained on, so on a real corpus the most likely output is a false positive against genuine footage. It is therefore built to be survivable when it is wrong — it never names, identifies or describes a person, it is advisory only, and it cannot reach a blocking decision. A finding says a file carries synthesis-like artifacts, never that a person is not who they appear to be. No accuracy figure will accompany this capability at any point — see the note on published detection figures in our engineering status. **Use case:** a recorded call is flagged as carrying synthesis indicators, prompting the verification procedure in capability 61 rather than substituting for it. **Risk:** the honest limit of any such model is that it recognises what it has seen; the control that does not depend on recognising anything is the one that stops the fraud.

Part 7 · Ransomware response and mobile endpoints (65–68)

Added 2026-08-28. Capabilities 65 and 66 close two gaps named in the entries above; 67 and 68 describe the mobile endpoints, which had no entry here at all despite an iOS build being in TestFlight.

65 · Recovery-path tampering detection — the ransomware precursor — Working · Teams: SEC

Almost every commodity ransomware family destroys the machine's own recovery path *before* it encrypts anything, so that paying is the only way back. The agent matches the command lines that do it: on Windows `vssadmin delete shadows`, `wbadmin delete catalog`, `wmic shadowcopy delete` and `bcdedit recovery-disable`; on macOS Time Machine snapshot deletion (`tmutil`, `diskutil apfs deleteSnapshot`); on Linux filesystem snapshot destruction (`btrfs`, `zfs destroy`, `lvremove`). Detection carries the offending process ID, which is what makes capability 66 reachable without guesswork. **Why it matters more than it looks:** this fires *seconds before* the encryption phase, so it is the earliest reliable warning the endpoint gets — and interrupting it preserves whatever shadow copies have not been deleted yet. **Honest limits:** the process monitor samples on an interval, so a command that completes inside one sampling window is detected after the fact rather than interrupted — which is why the durable paired response is host containment (44), not the kill. A match is discarded when a reader or version-control tool precedes the phrase, so searching a log for `vssadmin delete shadows` or committing a message that mentions it does not raise an alert, while a shell-wrapped real command still does. Reported by default; any automated response is tenant-armed exactly as in capability 39. **Use case:** ransomware lands on a workstation and begins clearing shadow copies; the endpoint reports it — and, if armed, kills it — before the first file is encrypted. **Risk:** the destruction of the recovery path is what converts an incident into a ransom negotiation.

66 · Process termination as an automated response — Configurable · Teams: SEC

Terminating the process responsible for destructive behaviour, through the same audited privileged-action broker as every other response (24). **This existed in the broker for months and could not be reached from the trigger that most needed it**, because a file event reports *what* changed and never *who* changed it — there was no process ID to act on. Two pathways now supply one: a recovery-tampering detection (65) carries the process ID natively, and mass-encryption/mass-deletion detections correlate it by finding the process holding files open in the directory being encrypted. **Honest scope:** the correlation is best-effort and says so — if the process cannot be identified, the detection returns no process ID and the kill declines rather than acting on a guess, while host containment (44), which needs no process ID, still fires. Off by default at the same three independent points as capability 39. **Use case:** the process encrypting a user's documents is stopped, rather than the whole machine being cut off the network. **Risk:** containment alone leaves the encryptor running locally; killing the wrong process on a workstation is its own damage, which is why a guess is never made.

67 · Mobile endpoint coverage — iOS — Pilot · Teams: SEC, AI

An iOS application carrying three network extensions: a packet tunnel that enforces AI-traffic policy at DNS/domain granularity, a Safari web extension that applies the same content pre-filter inside the browser, and an app-proxy provider for full-URL and request-body inspection. **Posture determines depth, and the app reports which it has rather than implying the better one:** on a supervised/MDM-managed device the CyberArmor certificate is pushed and trusted, so TLS is terminated on-device and the full URL and body are available; on an unmanaged personal device without that trust, flows are observed at SNI-hostname depth and

the reduced coverage is *reported as reduced*, never silently downgraded. Certificate authority is generated per device, and its private key never leaves the device. **Honest status:** in TestFlight, not yet in the App Store; the DNS-filtering and Safari paths are exercised, and the full-URL app-proxy path is built but requires device validation before it is switched on. **Use case:** a supervised tablet fleet where staff use AI assistants through a browser or native app, and the organisation needs the same policy on the phone as on the laptop. **Risk:** mobile is where AI assistants are used and where endpoint controls typically stop.

68 · Mobile endpoint coverage — Android — [Roadmap](#) · Teams: SEC, AI

A native Android agent — a [VpnService](#) enforcing AI-traffic policy on-device at DNS/domain granularity, with evidence flowing to the control plane. **Nothing has shipped.** The policy core is written as plain Kotlin with no Android dependency so it can be tested off-device, and it stays [Roadmap](#) until a build exists and has been validated on hardware. **A structural limit worth stating now rather than at pilot:** request-body inspection on Android is not achievable in any current mode, so Android coverage is planned at domain granularity, not payload. **Risk:** stating a mobile capability the product does not have is worse than the gap itself.

Release validation update — 2026-10-03

The public engineering status page remains the deployment-maturity reference.

The entries above also distinguish implemented/tested code from deployed client coverage. Maturity is reviewed from evidence; a green build does not automatically promote a feature.

- Signed macOS 1.0.7 PKG/DMG: production portal artifacts, with notarization, stapling, Gatekeeper and served-byte hash verification. Automatic agent binary upgrade is not implemented.
- Signed Windows 1.0.7 MSI/offline ZIP: published on dev and production with matching download hashes. Signature, timestamp and revocation checks passed; Windows also reports the MSI signature Valid. Fresh MSI enrollment, upgrade and uninstall remain rollout validation work.
- Local detection packs: configurable; five of five models executed after synthetic inference on macOS and Windows 11 Parallels VMs, with 20 of 20 pack files hash verified. Fresh-download interruption/recovery remains unvalidated.
- macOS AI inspection setup: listener, active proxy routing and TLS verified on the VM. Live Codex/Claude enforcement is not established by that setup check.
- Pakistan assessment mappings: production deployment; configuration checks, recorded AI activity and named documentary attestations are separate findings.
- Firefox/Safari 2.3.7: synchronized source and package checks; browser-store publication and fresh device validation remain separate release steps.