CyberArmor.AI Platform — Capability Definitions

84 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: 2026-11-22

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. **Payment-card and identity phishing (since 2026-10-28, on dev, production and Singapore):** a form that asks for a payment card number, CVV, expiry date, card PIN, one-time code or Pakistani CNIC counts as a credential form, not only a password field; a page that asks for a card PIN is phishing on its own, because no legitimate site asks for one; and 13 Pakistani banks are protected brands, matched only on their own domains, so a look-alike page carrying a bank's name and a card form on any other domain is blocked. Merchant checkouts and each bank's own site are not affected. Measured on the engine before and after the change (a fictional bank page asking for a card number, ATM PIN and CNIC went from allow to block); not yet measured in a browser against the deployed gate. Field labels in Urdu or other non-English languages are not read. **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. A bank's employee, or an AI agent working for one, follows a link to a page that copies the bank's name and asks for an ATM PIN; it is blocked before the PIN can be typed. **Risk:** indirect prompt injection — the attack class where the payload arrives in *content*, not from the user — and card-and-PIN phishing, which costs a bank its customers' money and its regulator's patience.

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 — 39-class catalog (36 structured + 3 NER) — `Working` · Teams: SEC, RISK
Data-loss prevention tuned for AI egress: 36 structured detector classes (SSN, credit card, bank routing, passport, driver's license, DOB, a debit-card PIN typed next to a card number, IP address, URL, crypto address, Pakistan's CNIC, NTN and mobile number, China's resident identity card number and mobile number, six PHI identifiers, 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. *(39 counted from the catalogue on 2026-10-29. The code recorded on dev and Singapore on 2026-10-28 holds 38; the card PIN class reached US production on 2026-10-29.)* **Honest scope:** a Chinese resident ID is recognised when it validates — province code, a real birth date and its check character — so a mistyped one is not; written with spaces or hyphens, or in the old 15-digit form, only next to a label such as 身份证; a number straight after another number's name – 订单号, 流水号, order no., payment ref – is read as that number, not an ID; the number is reported as a position and a length, never as its digits. 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. A link or an IP address on its own is a locator, not personal data (in the hosted AI router since 2026-10-28, on the endpoint from release 1.0.13-rc.15): it no longer makes a whole prompt count as containing personal data, so a "Redact PII"

policy stops blocking a coding agent's prompt because it holds a link, while a rule that names the URL or IP-address class still fires. 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 (logged only), 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. **Response scanning is best-effort and informational (2026-10-16).** A response is acted on only for a cited link the URL Trust Gate blocks and for prompt injection or promptware; sensitive data found in a response is logged, never blocked or redacted. When a response scan cannot run (a timeout, a 503, an overloaded local engine) the response is delivered and logged as `response_scan_unavailable_delivered` , whatever the tenant's fail mode; that setting still governs prompts and uploads (capability 15). Static and non-text responses (images, fonts, SVG, CSS, JavaScript, WebAssembly, and asset paths such as `/assets/`) are not response-scanned. **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 carrying a threat (a gate-blocked link, injection or promptware) 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` with documented format limits · 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, and neither is rewriting images, PDF or other Office files. When a redact policy matches a class it removes inside such a file, the tenant's `unredactable_file_action` decides: `warn` (the default since 2026-10-16) tells the browser user which policy and which data class and lets them choose **Upload anyway** (the original file is sent and the upload is logged) or **Don't upload**; the endpoint proxy cannot ask, so under `warn` it delivers the file, logs it as unconfirmed and shows a desktop notification; `block` refuses the attachment. 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. **Video and audio uploads** are read as well (53): video frames on the server and, on a Mac, on the device; audio is transcribed on the device from endpoint agent 1.0.13-rc.15. The spoken track of a video is not transcribed. **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. **From 2026-11-10 this reaches native desktop apps as well as the browser (#957):** the endpoint proxy now labels an upload as an upload, so an upload policy applies to a file leaving a desktop AI client exactly as it does to one leaving a browser tab. See 54 for what that fixed.

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. **Either way, the response says what was read** (built 2026-10-11, not yet deployed): the network proxy and the endpoint local proxy put an `x-cyberarmor-inspection` header on the responses they write. For the request and for the response, it states whether a content scan covered the body (`inspected`), covered only part of it (`partial`), was skipped by the local pre-filter (`prefiltered`), or was meant to run and did not, with the reason (`not_inspected:<reason>`). Before this, a request forwarded under fail-open looked identical to an inspected one. Some responses never pass through the proxy, such as the error a proxy returns when the provider is unreachable, so a response without the header must be read as unattested, never as inspected. **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. **Responses are the exception:** an unavailable response scan delivers the response, logged (capability 10).

16 · Endpoint / laptop agent coverage — `Working` (macOS/Windows/Linux; macOS kernel sensor Pilot; Windows user-mode ETW sensor Configurable — no driver, first run on a Windows test VM 2026-10-27, its fixes ship in 1.0.13-rc.15, 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, all built from one source; Safari live on iOS since 2026-09-24 with device counters read 2026-10-07, Firefox live on a production-enrolled Android tablet since 2026-09-27) · 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. **Re-validated on robot hardware 2026-10-07** (Raspberry Pi 5, robot purchased from Micro Center). Confirmed working: on-wire zero from the latching emergency stop, immediate hold, signed release, persisted latch, hardware stop request. Speed-limit enforcement verified on the wire (9.0 m/s command clamped to 2.0 m/s limit delivered downstream as 2.0 m/s). Telemetry reached the control plane. Not yet proven: control-plane **policy sync** — bootstrap-token retrieval was not completed in the test window; the safety layer enforces from local configuration as designed. **Enforcement is a deployment choice, and the default is observe.** With `actuator_output_topic` unset the agent evaluates every command and reports violations without altering what the robot receives; interposition (the mode validated on hardware) requires 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 nine operations:** apply patch, isolate host (which also releases), kill process, quarantine file, restore a quarantined file, update an isolation's release policy, and three proxy-configuration operations. **Honest scope, verified against source: apply patch** is reached from the control-plane command channel; **isolate host** is reached by automated response where the server has containment on (capability 44) and, in release mode, by a release command a tenant admin sends from the portal's Endpoints page, or by the endpoint itself when an isolation's release deadline passes; **restore a quarantined file** is reached by a restore command that names a quarantine item id and never a path; **quarantine file** is reached by the endpoint's own malware verdict (capability 43); **kill process** is reached by automated response when a tenant arms *Kill the process* (capability 66). **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:** each record is still signed with HMAC-SHA256 (symmetric). Since 2026-10-11 the audit service also signs a **checkpoint** over each tenant's chain with ML-DSA-87 (FIPS 204, row 32), every 1,000 records or 5 minutes: a digest of the full content of every record up to that point, verifiable with the public key alone, so an edited or reordered record no longer verifies. Each checkpoint is also appended to an anchor on a volume only the audit service mounts, so deleting the newest checkpoints is detected. Measured on production 2026-10-23: checkpoint signing enabled, the worker running, 0 chains failing. Root on the audit host can rewrite the anchor too; an external witness, such as an auditor's earlier export or a SIEM copy, closes that. Customers read and verify their own trail on the portal (80). **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; since 2026-10-28 the portal names each connector as its vendor writes it (QRadar, Google SecOps, Syslog CEF). Your SOC keeps its single pane; CyberArmor.AI 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. **Assessment on demand, and the last result on screen (since 2026-10-28):** the compliance page shows each framework's last stored result when it opens, with when it ran ("Last run") and over which evidence window, and nothing is assessed until someone runs it; on the Singapore region's portal that day, **Run all** assessed 45 of 45 frameworks. A risk

officer opens the page before an examiner meeting and sees where each framework stood and when it was last checked, rather than an empty page or a number recomputed without being asked. **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` (dedicated framework and pack shipped 2026-09-07) · 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, and, since 2026-10-11, the signed checkpoints over every audit chain (26), measured enabled on production 2026-10-23. **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.AI 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.AI source repository on 2026-10-07, and carried forward since in the release validation updates at the end of this document — the most recent of which, 2026-11-22, read the ninety-four changes merged since 2026-11-19 and states the limits they uncovered; the one before it, 2026-11-19, re-read the iOS and Android endpoint counters from both regions' control planes. 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. **Corrected 2026-11-13 (#958):** the portal's AI-traffic-proxy card read "Not installed, 0 reporting of 0 installed" on a fleet where every desktop agent was in fact running its own proxy, because nothing on that page read the two fields the agents had been reporting all along. It now counts the desktop agents' own proxies from `ai_traffic_status` and `ai_traffic_enforced`. A coverage card that undercounts is the same defect as one that overcounts: a number nobody can act on.

39 · Ransomware and mass-deletion: detection on by default, host containment off until a server turns it on — `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.

Killing the process can be armed today; host containment is built and waits for its server's switch. Host containment (capability 44) needs a way back, and that way back now exists: a tenant admin releases an isolated endpoint from the portal's **Endpoints** page, and each isolation carries the tenant's release policy. The release travels as a signed command, so an endpoint that could not verify it would stay isolated until someone reached the machine. Containment therefore stays off on a server until that server's operator turns it on, once endpoints pin that server's key; the agent also declines to isolate unless it can verify a signed release. While it is off, the control plane refuses a request to choose isolation and says why, and the portal does not offer it. **Settings → Automated endpoint response** — a tenant master switch (default off) and one row per detection — offers *Report only* (the default) or *Kill the process* (capability 66), and *Isolate the endpoint* where the server has it on. Detections are reported either way.

Where containment is on, how it arms and how it ends. 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*, and the agent's own local default, which stays off until it has successfully fetched the setting. A `tenant_admin` arms it by enabling the master switch *and* choosing "Isolate the endpoint" for a detection. A choice saved before 2026-10-11, when an isolated endpoint had no remote release, is not re-armed: it reads as *report only* until an admin chooses it again. **How an isolation ends is the tenant's choice:** *release automatically after N hours* (1 to 72, 24 by default) or *never — stay isolated until an administrator releases it*. The portal states the trade-off next to the choice: a deadline is something an attacker on a contained host can wait out; *never* leaves a host whose release command fails cut off until someone acts. A change applies to hosts already isolated at their next settings sync, counted from when each was isolated. A tenant admin can release a host sooner from the portal's **Endpoints** page.

The person at the keyboard is told (from endpoint agent 1.0.13-rc.15, Windows and macOS). One banner per incident says what actually happened: *CyberArmor.AI stopped ransomware*, naming the program and the folder, or *detected ransomware — NOT stopped* with the reason (the program had already exited, the kill was refused or failed, the program could not be identified, or automatic response is not turned on), and an instruction to contact IT. A user who sees it stops working in that folder and calls the help desk while it still matters, and the help desk hears the same outcome the portal records. Banners are in English only, and Focus or Do Not Disturb can hold them back; the status app says when it does (58). None has yet been seen on a test machine with this code.

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). **Windows endpoints are not isolated:** that path has never been executed on a Windows machine, so the agent refuses. **Recovery:** an isolated Linux host drops sshd's reply packets too, so the way back is the portal's Release or, unless

the tenant chose *never*, the release deadline, not SSH; both are built and unit-tested and have not yet been run on a device. **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 — `Pilot` · 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 through the privileged-action broker (capability 24) with its before-and-after audit record. A tenant admin can restore a quarantined file from the portal's **Endpoints** page by its quarantine item id: it goes back where it was with its owner and permissions — or beside it under a new name if something now has that name; nothing is overwritten — and for 30 days that endpoint does not quarantine the same file at that path again. Restore is built and unit-tested and has not yet been run on a device. **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 — `Roadmap` (built; off until a server's operator turns it on) · Teams: SEC
The automated response built for the capability-39 signals: the endpoint cuts its own outbound network access, permitting only the control plane, so it keeps reporting and can be told to release. IPv4 **and** IPv6 are both cut. **It is off on every server until that server's operator turns it on.** The release is a signed command, so a host that cannot verify it would stay off the network until someone reached it; the switch is turned on once endpoints pin that server's key, and the agent itself declines to isolate unless it can verify a signed release. While it is off, no tenant can choose it: the portal does not offer it, the control plane refuses it, and a previously stored choice reads as *report only*. **The way back:** each isolation's release policy is written before the firewall is touched. By default the endpoint releases itself 24 hours after the isolation began (1 to 72 per tenant); a tenant may instead choose

never, and then only an administrator's release ends it — the trade-off (a deadline can be waited out; *never* strands a host whose release fails) is stated next to the choice. An isolation the firewall shows but nothing recorded keeps a 24-hour safety deadline whatever the tenant chose. A tenant admin can release a host sooner from the portal's **Endpoints** page, which shows the state each endpoint measured from its own firewall, with since, trigger and deadline. A release reports success only when the firewall reads back as released, and a release command an endpoint has not fetched within 24 hours expires and is sent again from the same page. While isolated an endpoint checks in every 60 seconds, within a 4-hour daily budget, so a release is normally picked up within about a minute. Each isolation and release is recorded through the central audit sink. **Honest scope:** containment is verified by hand on Linux and, on macOS, read back out of pf before success is reported; release, the deadline and the portal path are built and unit-tested against simulated firewalls and **have not yet been run on a device**. **Windows endpoints are not isolated** — that path has never been executed on a Windows machine, so the agent refuses. 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. **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.AI 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.

Sizing consequence for every entry in this part: with no endpoint agent there is nowhere to run an on-device model pack (71), so agentless deployments get **no benefit from model packs**. Any AI traffic an agentless deployment inspects is inspected on the server or the hosted AI gateway, and detection must be sized for all of it; see the capacity planning guide, sections 4 and 6.

48 · Shadow-AI detection with nothing installed on the endpoint — `Pilot` · Teams: SEC, AI
The customer's existing sensors do the seeing. CyberArmor.AI 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.AI 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. **No on-device inspection:** nothing runs on these machines, so prompt and response inspection, where it happens at all, happens on the server or gateway (see the sizing note above). **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.AI 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.AI 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. **Video: the frames are read.** Frames are sampled and read with OCR, so text shown on

screen in a screen recording is found. The server reads them with its local OCR; on a Mac the endpoint reads them itself, decoding with PyAV and reading with Apple Vision (from endpoint agent 1.0.13-rc.13; measured on a Mac, a test clip was read on the device in 1.06 s and its text read correctly). Windows and Linux endpoints have no local OCR engine yet, so their frames are read by the server. A clip longer than the sampler's bound is read in part and reports how much of it was covered, never a clean scan. **Audio: transcribed on the device, from endpoint agent 1.0.13-rc.15.** On endpoints that run local models (71), an audio upload is transcribed by a Whisper speech model delivered in the signed model pack, inside the same sandboxed extraction process that parses documents, and the transcript is checked like any text: a card number or national ID read aloud into a voice note is found. Measured on a Mac only: a 12.5-second clip took about 1.8 to 2.1 seconds end to end, about 1.7 seconds of it a fixed cost per file. The server does not transcribe audio, so without rc.15 and its model pack an audio upload is held rather than passed, and the product says so. **The spoken track of a video is not transcribed:** only what is shown on screen is read, so an account number spoken in a screen recording is found only if it is also on screen. **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. A relationship manager uploads a recorded client call or a screen recording of the core-banking screen to an AI service to summarise it; the card number on screen, or spoken in the call, is found before the file leaves. **Risk:** DLP that only reads plain text is bypassed by the ways people actually move data — attach a document, screenshot it, record it, or say 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:** traffic from **containers** on the host is a separate surface with its own limits — see capability 55 and `docs/runbooks/docker-builds-behind-the-proxy.md`; the same runbook covers JVM, Go, Rust, Podman and WSL2 builds, which carry their own trust store and break the same way. 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 since run on Windows: on a Windows 11 VM with agent 1.0.8 (2026-10-04) the per-user proxy task listened on 127.0.0.1:8888, survived a stop/start cycle cleanly, and returned fail-closed policy decisions as configured. Inspecting a live AI desktop app end to end on Windows is the remaining step; that session's VM had no working outbound connection, so it could not be shown. **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. **From 2026-11-10 the proxy labels an upload as one (#957), and until then it did not.** It asked policy about every request carrying no surface label at all, so a `block_upload` policy was SKIPPED on every one of them (`policy_action_surface_skip … surface=<unset>`) and a file the browser extension would have stopped left the ChatGPT desktop app, Claude Code or Codex unimpeded. The proxy now sets `request.type` to `upload` or `prompt` and carries the browser extension's exact vocabulary — `content.has_file_upload`, `file_count`, `file_names`, `file_extensions`, `file_types`, `file_bytes` — so one rulebook decides a desktop upload and a browser upload the same way, and those seven fields are resolvable by the server engine too. A body nobody could read is recorded as unread, never as a body with no files in it.

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`. **A fifth**

authority was added on 2026-09-21: traffic from containers on the host (`container_egress_inspected`). Docker Desktop auto-detects the endpoint's system proxy and routes container egress through it, but only on the ports in its own transparent-port list — **TCP 80 and 443** as shipped — so a container reaching a model endpoint on any other port is uninspected, and that port list is printed as the denominator. **Since 2026-10-10 this is measured in the user's session, not by the root agent:** a small per-user agent reads Docker's settings and asks each container engine over its own socket (Docker Desktop, OrbStack, Rancher Desktop, Colima, Lima, Podman), and hands the result to the root agent as *user-reported*. The root agent runs no Docker software and opens no socket of the user's. Because the user could forge it, the check is **attestation-only and cannot be verified by the privileged agent**: a user-reported gap is shown as a gap, and a user-reported "routed to us" (or "nothing to inspect") is shown as **not verified** — it keeps the endpoint from reading as fully enforced; a missing or stale report reads as *could not be checked*, never as covered. An engine that is installed but whose socket cannot be located reads as *not known*, not as stopped. **Honest scope:** a container verifies TLS against its own image trust store, so interception makes it fail closed; builds that fetch from the network break unless the certificate authority is supplied to that build. **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. **From 2026-11-10 (#960) an app that does not trust the proxy's CA is named, not merely counted:** the client is attributed to a PROCESS from its port, that process is told once to restart, and it is listed with the CA file's SHA-256 and modification time so an operator can tell a stale trust store from a missing one. The entry clears when that same process completes a handshake. A client the proxy could not attribute is counted rather than dropped — an app it cannot see into has to be a number the operator can read, not a silence.

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. **Measured on the Singapore region's portal, 2026-10-28:** an attestation with a policy document attached raised that tenant's NIST CSF score from 33% to 39%. Two display defects were seen there and are not yet fixed: the Attested count stayed at 0 after the attestation, and the control's row said "no note" without naming the attached file. **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 no disable or uninstall in it.** On macOS it offers a time-boxed Pause of AI-traffic inspection, available by default since 2026-10-16 so a false positive never strands legitimate work; organizations turn Pause off through device management (MDM/GPO) or the tenant setting, which can also set a lower maximum; every pause lasts 5 to 120 minutes (30 by default), is reason-prompted with an optional finding id, audited, and resumes on its own. **Since 2026-10-22 (built and unit-tested, not validated on hardware):** the Windows tray offers the same Pause – the same organization switches and durations, started through the Windows administrator (UAC) prompt, resumed by the agent at the deadline, audited – and on macOS the menu bar, heartbeat and portal name each watched folder that needs Full Disk Access and how to grant it, never reporting a folder that was not checked as watched. The only exit hides the icon until next login and never stops the protection. **From endpoint agent 1.0.13-rc.15 (built and unit-tested):** a **Show notifications** switch, on by default and kept per user, through which every banner the status app posts goes; and a row beneath it that says

when the operating system is holding banners back (Focus or Do Not Disturb, or notifications turned off for the app), or that it could not check — it can open the settings page and never changes a system setting. The same release fixes macOS banners from the status app that were being lost because the app was not registered under its own identity, which was found while filming. Banners are English only. 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. **From 2026-11-10 (#959) the person sees a flag, and the counts have a list behind them.** A `warn` verdict dispatches a desktop banner, and the proxy keeps a durable record beside its health file of every block, every flag and every redaction that was actually applied — ten fields each (when, kind, host, path without its query, policy, reason, cause, who decided, the detection class names, and the request id), a 200-entry ring that says what it dropped and carries lifetime totals across a restart. The status menu shows the three counts and each one opens its own list; both platforms’ command lines take `--list flagged|blocked|redacted`. **The wording here is deliberate and load-bearing:** a dispatched banner is not a delivered one, and there is no API on that path to ask whether it was displayed, suppressed by Focus, or denied. So the shipped noun is **flagged and let through**, counted at `enforcement_health.json:period.warnings.request`, and nothing on any surface claims the person was warned.

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.AI 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). A file event reports *what* changed and never *who* changed it, so the process comes from the operating system's own sensor that watched the change: **Endpoint Security on macOS and ETW on Windows** record the process (ID, start time, executable, and signer on macOS) for each write, rename and delete in the user's Documents, Desktop and Downloads, and the mass-encryption and mass-deletion detections (39) carry that process or say *unattributed*, with the reason. A recovery-tampering detection (65) carries its process natively. A tenant administrator chooses **Report only** (the default) or **Kill the process** per detection. **Windows, measured on a Windows 11 test VM on 2026-10-27** with the fixes that ship in endpoint agent 1.0.13-rc.15 applied by hand: before them the Windows sensor decoded no event property and every kill was refused; with them the ransomware simulator was killed about 2 seconds after detection, after 7 of 40 demo files were encrypted on an rc.12 build and 6 of 40 on an rc.13 build, and the portal recorded the kill with outcome *killed*. The kill setting survives an agent restart, a second

attack inside the alert cooldown is still stopped, and a process that had already exited is reported as *not stopped*, never as killed. The person at the keyboard sees a banner saying which (39). **Honest scope:** macOS is built and not validated on hardware; Windows has run on one test VM, not a customer endpoint, and not yet from a released build. The kill runs only if the live process still has the observed start time and executable, so a reused process ID is never killed; the agent, its sensors and system-critical processes are never killed; Linux endpoints have no sensor and are reported only. The earlier best-effort correlation (the process holding the most files open in the directory) was removed, because it was a guess. Detection alone does not stop encryption, and files already changed stay changed. Host containment (44) still cannot be armed. **Use case:** the process encrypting a user's documents is stopped, rather than the whole machine being cut off the network: on the test VM, 33 or 34 of 40 files were never touched. **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 — `Configurable` · 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.AI 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. **Live counter evidence, re-read from both regions' control planes 2026-11-19:** the US control plane holds eight iOS and iPadOS registrations, the newest on iOS 27.2.1 (agent 1.0 (27)) reporting 6,227 AI lookups observed and 72 blocks in its last 24 hours; Singapore holds a ninth, on iOS 27.2.1 (agent 1.0 (26)), reporting 4,951 lookups and 48 blocks. Earlier reading 2026-10-07: DNS filter active, policies enforcing, blocking and logging events accumulated since 2026-09-24; Safari extension counters confirmed on same device — pages checked, phishing blocked, sensitive data redacted, warnings shown, findings logged; no issues reported over two weeks. No new Safari counter was read on 2026-11-19, so that sentence is unchanged. Enrolled across multiple production tenants. **Honest status:** in TestFlight, not yet in the App Store. The full-URL app-proxy path is built and, since 2026-10-07, wired into the app; running it on a device under MDM is the remaining step (capability 69). **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 — `Configurable` · 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. **Live hardware validation completed:** APK enrolled and running on physical hardware in multiple production tenants; production audit records confirm device connection and live counter accumulation. The policy core is plain Kotlin with no Android dependency so it can be tested off-device. **A structural limit worth stating:** request-body inspection on Android is not achievable in any current mode, so Android coverage is at domain granularity, not payload. Work-profile inventory is not started. **Risk:** payload inspection is not possible; domain-level enforcement is what ships. **Counters re-read from both regions' control planes 2026-11-19:** Singapore holds two Android 16 registrations, the newer on agent 1.0.2 reporting 5,226 AI lookups observed and 4,807 blocks in total; the US control plane holds a third, on agent 0.1.0, reporting 2,160 lookups and 2,112 blocks.

Part 8 · Expanded platform capabilities (69–72)

Added 2026-10-09. Four capabilities: the iOS App Proxy for full request-body inspection on managed fleets, the ROS 2 robotics policy agent as a standalone entry (it was a footnote in capability 22), on-device AI detection model packs for air-gapped and offline enforcement, and the voice-triggered robot stop now in development on top of the ROS 2 agent.

69 · iOS App Proxy — full URL and request-body inspection on managed devices — `Roadmap` (built and wired; MDM device validation next) · Teams: SEC, AI
The third layer of the iOS endpoint: where the DNS-level packet tunnel (67) enforces at hostname granularity, this provider sees the

full URL path and request body — which means it can distinguish an ordinary chat from an exfiltration POST to the same AI host. **How it works:** `NETransparentProxyProvider` captures every AI-bound flow. On a supervised or MDM-managed device the CyberArmor.AI CA is pushed and trusted silently; TLS is terminated on-device, and the full URL, headers, and body pass through `FlowPolicyEngine` before the bytes leave. On an unmanaged (BYOD) device the same provider runs but operates at SNI depth, and that reduced coverage is reported explicitly in the app — never silently implied as full inspection. The per-device CA private key never leaves the device. **Wired and active 2026-10-07** (commit `746fb088`): `AppProxyController` is instantiated in the app lifecycle and reports its coverage as `running`, `notInstalled`, `blockedByTrust`, or `unavailable`, so the UI always reflects actual coverage rather than a nominal claim. **What this means for a managed fleet:** a corporate tablet policy that routes AI traffic through the proxy can inspect and enforce on the full request body, redact secrets before they reach any model, and record every AI interaction with URL-level precision. **Honest status:** built, unit-tested, and wired into the shipping app; the one remaining step is running it on a device under an MDM trust profile and reading its counters, after which it is promoted — mobile capabilities move up a tier only once a device has run the code. **Use case:** a hospital fleet of managed iPads where staff use AI assistants; the full request body is inspected for PHI before it leaves the device. **Risk:** without full-URL inspection, domain-level controls can be evaded by routing sensitive data to a path on an approved AI host.

70 · ROS 2 robotics policy agent — `Pilot` (hardware-validated 2026-10-07) · Teams: SEC, AI
A purpose-built AI safety and audit layer for ROS 2 robots: every actuator command passes through a policy gate *before it reaches the motor driver*. This is the control that matters in robotics — a policy enforced at the message bus, not observed after the wheel has already turned. **What ships:** topic graph monitoring and DDS participant inspection (know what is on the bus and what has changed); actuator policy in observe mode (record every command against policy, flag violations) and interpose/enforcing mode (rewrite or suppress a violating command before it is published downstream); a **latching emergency stop** that persists across restarts and requires a signed release token to clear — no code path can clear it with a plain boolean; software motion inhibit; and sensor integrity monitoring. **Re-validated on robot hardware 2026-10-07** on a Raspberry Pi 5 running a real robot purchased off the shelf. Confirmed on the wire: the latching e-stop produced zero downstream output; speed-limit enforcement clamped a 9.0 m/s command to the 2.0 m/s policy limit, delivering 2.0 m/s to the motor driver; telemetry reached the control plane; persisted latch survived and was released by signed token. **Air-gap ready by design:** the safety layer enforces from local configuration when no control plane is reachable — a disconnected robot is not an unenforced one. Remote policy sync from the control plane is on the roadmap for connected fleets. **Command text (76, built 2026-10-13, not yet on robot hardware):** the agent now also inspects the *text* that drives a robot (voice-command transcripts, LLM and VLM planner steps and requests from robot apps) on the robot itself, from the same rules engine and the tenant's signed policy bundle; MCP tool calls are next, on the roadmap. Device read and write commands are planned to be governed in a way compatible with Anthropic's Model Hardware Standard, pending its published specification; no conformance is claimed. **Use case:** an autonomous forklift fleet where an AI agent controls navigation; a single hallucinated 9 m/s command in a loading dock is caught and suppressed before it reaches the drive train. **Risk:** without an agent between the AI and the actuator, the only safety layer is the model itself.

71 · On-device AI detection model packs — offline and air-gapped inference — `Configurable` · Teams: SEC, ENG
Signed, AES-256-GCM encrypted detection model packs that run entirely on the endpoint — no data leaves the device during inference. Designed for air-gapped environments, high-security deployments, and networks where sending AI traffic to a central detection service is not permitted. **How delivery works:** packs are signed by the control plane and delivered via ML-KEM-1024 key exchange (post-quantum); the endpoint unwraps and verifies the pack before writing it to the local store; an atomic swap with automatic rollback ensures a corrupted or interrupted download cannot leave the endpoint with a partial model. The local engine evaluates prompts at three tiers: rules only, rules and models, or rules, models, and threat feeds — tenants choose the depth. **Verified 2026-10-03:** five of five models executed after synthetic inference on macOS and on Windows 11 (Parallels VM), with 20 of 20 pack files hash-verified. The extension also routes inline browser prompts through the local engine as the user types, not only at submit — so detection happens before the prompt is sent. **Uploaded documents are scanned on the endpoint too** (2026-10-06): text, DOCX, XLSX and PPTX attachments are extracted and inspected locally before upload, so a busy detection service no longer turns into a blocked upload; formats that need OCR or a PDF parser (PDF, images, legacy Office) were routed to the server-side extractor with offline OCR (capability 53) rather than half-read on the device. Since then the device parses PDF and legacy Office itself (74), and on a Mac reads images and scanned pages with Apple Vision. **Responses as well as prompts** (2026-10-10, built, not yet run on a device): the pack's output-safety model is now a small distilled classifier, loaded at start, so an AI *response* can be checked on the device instead of only the prompt. Packs built before this change carry the retired model and must be rebuilt; until then the endpoint reports that check unavailable, never clean. **Speech as well as text** (from release 1.0.13-rc.15): the

pack carries a Whisper speech model (base) so audio uploads are transcribed on the device (53); its weights are decrypted into memory and handed to the sandboxed extraction process, never written to disk in plaintext, and the pack that carries it is offered only to rc.15 agents. **Each endpoint reports the share it decided itself** versus sent to the server (`local_inspection` in its heartbeat, with the count it is a share of), so server capacity is planned from the fleet's real ratio rather than an assumption. **Agentless deployments do not benefit:** packs run on the endpoint agent, so a deployment with no agent inspects everything on the server or gateway (Part 4). **Current boundary:** fresh-download interruption and recovery on a live fleet remains unvalidated; the infrastructure is built and the rollback path exists but has not been exercised end-to-end on physical hardware with a real pack download. **Use case:** a defence contractor on an air-gapped network who needs the same prompt-injection and data-exfiltration detection as a cloud-connected deployment, with no traffic leaving the facility. **Risk:** cloud-only detection means a network outage or policy against external traffic leaves the endpoint unprotected; local packs keep enforcement running regardless.

72 · Voice-triggered robot stop ("ROBOT STOP") — `Roadmap` · Teams: SEC, AI
Anyone near a robot can stop it by saying so. A spoken stop phrase, recognised on the robot's own companion computer with no network involved, becomes one more input into the ROS 2 agent's latching software stop (70) — the same latch, the same signed-token release, the same audit record. **Built so far:** a multilingual stop-phrase set (English, Urdu, Mandarin and Spanish in v1) delivered as an Ed25519-signed, versioned artifact, so the phrases a robot listens for are as controlled as its policy; a bench-test harness that reports false-reject and false-accept rates and latency percentiles per language; and the keyword-spotting engine, openWakeWord (Apache-2.0, fully offline, CPU-only), wired into that harness. **Design choices that matter to a safety team:** recognition is a bounded keyword spotter, not open-ended transcription, so it is small, fast and auditable; the threshold is tuned toward catching every real stop, because a missed stop is the outcome the control exists to prevent; and voice is an *additional* way to trigger the software stop, never a replacement for a physical e-stop — ISO 13850 does not recognise voice as a conforming actuation means, and we do not claim it is one. **Next steps:** train per-language phrase models, English and Mandarin first (decided 2026-10-12; Urdu recording is on hold), measure them on recorded human speech under real ambient noise, and wire the detector into the ROS 2 agent. **Use case:** an operator with both hands full on a warehouse floor shouts "Robot stop" and the arm latches before anyone reaches a button. **Risk:** in a shared human-robot workspace, the nearest stop button is not always within reach — the voice is.

Part 9 · Local-first decisions and new coverage (73–76)

Added 2026-10-13. Four capabilities merged between 2026-10-12 and 2026-10-13 and deployed to the hosted stack (dev, production and Singapore) and to the endpoint agent 1.0.12 beta, the browser extension 2.3.11 and the ROS 2 agent 0.2.0. Each is `Configurable`: built, tested in continuous integration and released, and not yet measured on a customer device or on robot hardware. None moves up a tier until it has run there and its counters have been read.

73 · Local-first decisions on the endpoint — `Configurable` (shipped in endpoint agent 1.0.12 beta and extension 2.3.11; not yet measured on a customer endpoint) · Teams: SEC, AI
The endpoint decides for itself. When its local engine is healthy, the URL Trust Gate verdict, file scans and the tenant's policy are decided on the device, and the backend is used for detonation, policy sync, telemetry, audit, updates, response, confirmations and status. **How it works:** policy comes from the tenant's signed, expiring policy bundle, verified on the device (signature, tenant, expiry, anti-replay) before anything evaluates it; the keys that sign it can be carried, rotated and revoked by the tenant's signed posture. A local allow from the URL Trust Gate needs a fresh, verified reputation answer; without one, the allow goes to the server. A verdict the engine recommends for detonation goes to the backend, and if the backend does not answer, the local verdict stands and says so. The engine runs under launchd on macOS, as a SYSTEM task on Windows and, new in this release, under systemd on Linux. The proxy authenticates the local engine before it trusts a local answer. **Every answer names who decided it** (`decided_by`, the tier, and anything it decided without), and each endpoint reports the share it decided itself. **Honest status:** on a Windows 11 VM the 1.0.12 MSI installed, the local engine ran and answered its health check with its token and refused without it (2026-10-13). Not yet measured on a customer endpoint. **Use case:** a hospital laptop keeps enforcing policy during a network outage, and the backend sees only what it must detonate. **Risk:** a central decision service is a single point of failure and sees every request.

74 · On-device document parsing — `Configurable` (shipped in endpoint agent 1.0.12 beta and extension 2.3.11; not yet measured on a customer endpoint) · Teams: SEC, RISK
The endpoint's local engine reads PDF, Office (DOCX, XLSX, PPTX), legacy Office (DOC, XLS, PPT, MSG), RTF, email and archives (zip, tar, gzip, bzip2, xz) itself, with the same parser package the detection service runs. It opens attached and embedded files, and flags active content: PDF JavaScript and launch actions, VBA and XLM macros, ActiveX, DDE, external templates and OLE packages. A file with active content, and a format it does not open (7z, RAR, CAB, executables), is sent to the backend for detonation. Images and scanned pages are read on the device on a Mac, with Apple Vision; Windows and Linux have no local OCR engine yet, so those go to the server-side extractor with offline OCR (53). Video is decoded on the device from release 1.0.13-rc.13, and audio is transcribed on the device from rc.15 (53). Nesting depth, member counts and inflated size are bounded, and each bound is named when it applies. **Honest status:** tested in continuous integration; not yet measured on a customer endpoint. **Use case:** a contract PDF dragged into an AI chat is read and checked for regulated data on the laptop, without being uploaded for inspection. **Risk:** a document the endpoint cannot read either leaves unchecked or has to be sent off the device to be read.

75 · Chinese AI services recognised and inspected — `Configurable` (deployed to the hosted proxy 2026-10-13; in endpoint agent 1.0.12 beta and extension 2.3.11; not yet run against live traffic to these services) · Teams: SEC, AI
Prompts to twelve Chinese AI services, alongside DeepSeek, are recognised as AI traffic and reach the scanner: Moonshot Kimi, ByteDance Doubao, Alibaba Qwen, Baidu ERNIE, Zhipu GLM, Tencent Yuanbao, iFlytek Spark, MiniMax, StepFun, Baichuan, 01.AI and Metaso. They are in the shared AI service catalog, so Shadow AI discovery counts them, and both proxies and the browser extension inspect them. Prompts written without spaces, as Chinese is, are read as text rather than dropped as tokens. **Honest status:** each service's web and API hosts were checked to answer on 2026-10-12, and request shapes are covered by tests on both proxies and in the extension harness. Not yet run against live traffic to these services. **Use case:** a regional team uses Kimi and Qwen; a resident ID card number in a prompt is recognised as a sensitive data class (capability 8), and the browser extension and the endpoint agent redact it under a policy that names that class, as the China and HKMA packs do. On its own it does not block the prompt. **Risk:** an AI service the platform does not recognise is a blind spot, whatever the policy says.

76 · ROS 2 on-robot command decisions, with human notification — `Configurable` (in ROS 2 agent 0.2.0; tested in continuous integration; not yet run under ROS 2 on the Raspberry Pi) · Teams: SEC, AI
The robot decides what to do with the text that drives it. Voice-command transcripts, LLM and VLM planner steps, and requests from robot applications are inspected on the robot, with the same rules engine and policy evaluator as the services and the endpoint, and the tenant's signed policy bundle verified on the robot. Each command is allowed, redacted, held for a human to confirm, or blocked, and the strictest answer wins. **Failing closed:** a command the agent cannot inspect is not forwarded; that includes an inspection fault and text over the 1 MiB ceiling. When no policy bundle is usable, the rules verdict stands and the verdict says the policy tier was not available, or, if the operator chooses, everything is held for a human. **Every dropped command is reported to a human with its reason:** blocked, held with no answer before it expired, evicted from a full queue, too long, or an inspection fault. Each one is published on the robot, raised as a critical event and sent as telemetry, and the operator can also put the robot into its safe state on a drop. A held command is released only by a named local user or a signed, single-use token bound to the exact text held. Applications ask through a local socket that admits callers by their operating-system identity. Node identity uses DDS endpoint identifiers and an SROS2 enclave per agent; the agent reports whether SROS2 is enforcing for its enclave and does not claim it when it is not. No command model ships yet: decisions come from rules and signed policy. Needs Python 3.11 or newer (ROS 2 Jazzy or later); on Humble the agent runs without command inspection and says so. **Honest status:** built and tested in continuous integration with stand-in ROS messages. Not yet run under ROS 2 or on the Raspberry Pi 5, and its counters have not been read on a robot. **Use case:** a warehouse robot's planner emits a step that came from an injected instruction; it is held for the operator instead of executed, and the operator is told why. **Risk:** a robot that executes whatever text reaches it inherits every prompt-injection weakness of the model that wrote it.

Part 10 · AI coding agents, MCP and the customer's own record (77–80)

Added 2026-10-23. Four capabilities merged between 2026-10-18 and 2026-10-23. The server halves are in production's checkout at `262f597`, whose 22 running service images matched it on 2026-10-23; the endpoint halves are in the endpoint agent 1.0.13 release candidates. None has been measured on a customer endpoint, and each sits at the tier that says so. Entries 78 and 80 were updated 2026-10-28.

77 · MCP tool-description protection — tool poisoning and rug pulls — `Roadmap` (built, not validated: unit-tested; not yet run with a real Claude Code or Codex session or on a customer endpoint) · Teams: SEC, AI, ENG
An MCP server's tool list is handed to the model verbatim, so a malicious or compromised server can hide instructions in a tool's name, description or schema (**tool poisoning**), or change a definition after it was approved (**rug pull**). This removes a poisoned tool from what the model is given. A tool is hidden when its definition carries text hidden from people (tag characters, zero-width or bidirectional characters inside words, HTML comments, hidden markup) or names another server's tools; the payload is decoded and read by the same prompt-injection classifier the platform already runs, and when the text reads as instructions to the AI the tenant can warn or alert. Each tool's definition is hashed (the same hash in MCP, Anthropic and OpenAI shapes); a tenant approves a server's tools by hash, and a tool whose definition changed afterwards is treated as unapproved until it is approved again. **Where it runs:** both proxies (every model request's tools, which covers local stdio servers, and the tools/list answer on hosts the proxy decrypts), the Claude Code and Codex hooks (a call to a hidden tool is denied), and the Python SDK. A definition that could not be scanned is reported as not checked and follows the tenant's fail mode; it is never cached as clean. **Honest status:** unit-tested only. There is no portal screen yet, it has not been run with a real coding assistant, and the hosted proxy applies one policy to all its tenants from configuration. **Use case:** a developer installs a community MCP server whose `search` tool description quietly tells the model to read `~/.ssh` and paste it into the next call; the model never sees that tool. **Risk:** the tool list is the one part of an agent's context nobody reads, and it is read with the authority of a system prompt.

78 · The Trust Gate and tenant policy in front of AI coding agents (Claude Code, Codex) — `Configurable` (in the endpoint agent 1.0.13 release candidates; the hooks are installed per user; not yet measured on a customer endpoint) · Teams: SEC, ENG, AI
Local AI coding agents fetch web pages and read files on their own. Before Claude Code or Codex fetches a page (a web-fetch tool, or a shell `curl`, `wget`, httpie or `python -c` fetch), the URL goes to the URL / Context Trust Gate (1), on the device's local engine first: a page the gate blocks is denied before its content reaches the model, naming the threat, and a page the gate would redact can be read through a bundled **safe-fetch** MCP server that returns it with the hidden content removed. Before the agent reads a file (its read tool, an `@file` mention in the prompt, or a shell `cat`, `head`, `sed -n`, `Get-Content` or `python -c open(...)`), the file is scanned (PDF, DOCX, XLSX and text, up to 10 MB) and evaluated against the tenant's upload and file-read policy: block denies the read and names the policy and the data classes; redact allows it with a logged warning, because an agent's read cannot be redacted (a tenant that wants it stopped sets block). A gate that does not answer follows the **tenant fail mode** (15). A file that was too big, unreadable or had no detector is allowed **and said to be unchecked**, never recorded as clean. Telemetry carries scheme, host and path only, never the query or the page. **From release 1.0.13-rc.15:** a **video, screenshot or scanned PDF** the agent is given is read before the agent reads it — on a Mac on the device with Apple Vision, on Windows and Linux by the server — where before it was allowed and said to be unchecked; the bundled safe-fetch is no longer denied as if it were a raw fetch; and a URL handed to a tool is no longer scored as a prompt injection (the classifier had been reading demo page file names as sentences). **Honest status:** built and released in release candidates. The rc.15 changes were measured in-process on a Mac with the real hooks and engine on 2026-10-27: a screen recording showing an SSN and a card number was denied to both Claude Code and Codex, its clean twin was allowed, and a screenshot and a scanned PDF carrying an ID were denied. Not yet run in a real Claude Code or Codex session, on Windows, or on a customer endpoint, and the Windows hook installation is still being validated. **Use case:** an engineer asks the coding agent to summarise a vendor's documentation page; a hidden instruction on that page never reaches the agent, and the customer contract the agent tries to read next is stopped by the same policy that stops it being pasted into a chat. A developer at a bank points the agent at a screen recording of a production incident; the customer account numbers on screen are found before the agent reads the frames. **Risk:** a coding agent with shell access reads whatever it is pointed at, with the developer's permissions.

79 · Clipboard redaction for native desktop AI apps — `Pilot` (macOS, and the Windows helper validated on hardware 2026-10-24) · Teams: SEC, RISK
A native desktop AI app never crosses the browser, and text pasted into it may never cross a proxy either. A small helper runs in each user's own session, evaluates what is copied through the same policy client as every other surface, redacts it in place on a redact decision, clears it on a block, and reports telemetry either way. Redaction is scoped to the classes the policy decision names, so a prompt-PII policy redacts the same classes on the clipboard as in a chat window. **Where it runs:** on macOS as a per-user LaunchAgent; on Windows as a per-user logon task, with a tray row that says whether the helper is running, not running or not installed (that row could not measure the helper on any Windows install before release 1.0.13-rc.15; fixed from rc.15). **Honest status:** macOS is what has run; the Windows helper was validated on hardware 2026-10-24: a Windows endpoint on the pk-

demo-bank tenant reported four detect-then-redact pairs between 17:08 and 18:51 UTC, each about a second apart with no problems reported — pii.pk_cnic alone, pii.credit_card alone, and twice pii.pk_cnic with pii.pk_mobile, pii.credit_card and pii.iban — each redaction rewriting the clipboard (96→99, 505→502, 46→49, 381→378 characters). Four clipboards on one machine in one session is not a fleet, so the tier stays Pilot. **Use case:** an analyst copies a customer table with account numbers and pastes it into a desktop AI client; what arrives has the account numbers replaced. **Risk:** the copy-paste path is the oldest data-loss path there is, and the one a network control cannot see. **From 2026-11-04 (built, not validated on hardware):** for what is *typed* into such an app, and for a paste at the keystroke, the helper also carries a native-app guard: the focused text field of the foreground application is read, read-only, through the operating system's accessibility API and checked against the same policies, with a notification naming the app, the classes and the policy when a warn, redact or block policy matches, and the clipboard rewritten with the engine's placeholders at Ctrl+V / Cmd+V when a redact policy covers it. A block policy cannot stop a send inside a native application; the guard says so rather than claiming it. Not yet run on a device.

80 · The customer's own signed audit trail, readable and verifiable per record — `Pilot` (in production's portal checkout; opened on the Singapore region's portal 2026-10-28) · Teams: RISK, SEC
The signed, hash-chained trail (26) gets its own screen in the customer portal, scoped to one tenant and read-only; there is no write path. Each record shows its time (UTC, labelled), actor (user, API key, device, agent, service, operator or partner, with "via"), action, target, outcome and source address (or "not recorded"), with a **per-record verification badge**. The detail view shows the record's own bytes. A customer can export the **original records** (JSONL) or CSV, and run **Check the whole trail** for forks and the ML-DSA-87 checkpoints. The screen that was previously called "Audit Logs" read the control plane's own unsigned API access log; it is now labelled **API access log** and says it is not signed. **Honest status:** merged 2026-10-18 and in production's checkout. Opened on the Singapore region's portal on 2026-10-28, where **Check the whole trail** found no forks and verified the ML-DSA-87 checkpoints over 5,131 records in 1 chain. One tenant on one region, so the tier stays Pilot. **Use case:** an examiner asks for every policy change in March; the compliance officer exports the original signed records and the examiner verifies them without trusting the vendor. **Risk:** evidence the customer cannot read for itself is the vendor's word.

Part 11 · Secure by default, in the customer's language, and the money's destination (81–83)

81 · Protection levels — one step from purchase to protected, measured rather than asserted — the hosted control plane and customer portal `Working` (deployed and measured on production and Singapore 2026-11-08); the per-device view `Configurable` (built and shipped in endpoint agent 1.0.18 and browser extension 2.3.25, not yet read on a customer device) · Teams: SEC, RISK
Most organisations that buy an AI security platform do not know which of its controls to turn on, and those controls are exactly what decides whether anything is protected. A **protection level** is the whole posture chosen in one step: five of them, most to least protective — **High AI Security & Trust**, **Strong Protection**, **Balanced Protection**, **Light Protection** and **Monitor Only** — each a complete, named set of values across the 28 tenant settings the portal can write, with the server's own one-line reason per setting rendered on the page. A tenant nobody has chosen for gets the most protective level, and a first-login questionnaire asks three questions, recommends a level, says what the recommendation does *not* include, and applies it with one button. The granular pages stay exactly as they were, as the **Advanced** environment; no control here is new. **It is a measurement, not a stored choice.** Asked for a tenant's level, the platform reads every value back through the same getters the Advanced pages use and compares them: the badge says the level only when every compared setting matches; when one differs it says **Custom** and names the setting, what was expected and what is actually set; when one could not be read it says **unverified** — never a level, and never Custom, because an unread setting is not a changed one; and when nobody has chosen, it says that instead. Every answer carries its denominator ("9 of 9 settings for this computer match"), and every customer write route the control plane declares is either mapped to a level or excluded with a recorded reason, enforced by a test that fails on a new unlisted route, a stale exclusion, or a setting naming a route that does not exist. **Every surface says the same thing, in the server's words.** The same measured answer reaches the customer portal, the partner console (where a partner administrator can see and set a managed tenant's level), the macOS panel and menu, the Windows tray, the Linux status command, the browser extension's popup and options page, the Office task pane, the VS Code, Cursor, Kiro and Visual Studio status views, the iPhone, Apple Watch and Android screens, the ROS 2 robot's status topic, and the endpoint's own AI-traffic proxy — each showing the level, what is protected right now, what is

not set up, and **one** next step. No device spells a level name or a mapping value of its own, which a test asserts, so a device can never disagree with the server about what the organisation chose; a device the tenant has permitted to pause shows **Custom** with the count rather than the level; and the hosted gateway, the AI router and the proxy agent report that they read no level rather than implying one. **The most protective level still lets people work:** private network addresses warn rather than block, Pause stays available, the advisory media models stay off, and automated response only ever kills a process the operating system's own sensor attributed. Fail mode is closed at the two highest levels and open below, and that tolerance is what moves a tenant between them. **Honest status:** the route and the portal view are deployed. Measured read-only on production and Singapore on 2026-11-08 — both control-plane images carry `protection_levels.py`, `GET /api/customer/protection-level` answers 401 unauthenticated rather than 404, and the portal serves the **Protection** view with its `/protection-level/` route allowed through nginx. The device-side answer is built, tested on every surface listed above and shipped in endpoint agent 1.0.18 and browser extension 2.3.25; **no customer device has been measured reading a level**, the ROS 2 node has not run under ROS 2 on a robot, and the endpoint proxy's reporting has not run on an installed endpoint. **Use case:** a company buys three seats on Friday and is protected on Monday without reading a control reference — and its auditor can be shown, setting by setting, that the posture the company chose is the posture its machines are actually in. **Risk:** a security product whose defaults are permissive, or whose dashboard reports a posture it never verified, protects nobody while appearing to.

82 · The product in the customer's own language — eleven display languages — `Working` (eleven languages live on the marketing site, both portals and this documentation, measured on production 2026-11-08; ten of the eleven are machine-drafted and say so on every page) · Teams: RISK

Everything a customer reads — the marketing site, the customer portal, the partner console and this documentation — is offered in **eleven display languages**: English, Simplified Chinese, Brazilian Portuguese, Latin American Spanish, Korean, Japanese, Thai, Vietnamese, Indonesian, Malay and Filipino. One registry is the single list of them, and the display runtime takes however many languages that registry holds rather than carrying hard-coded ones, so a twelfth language is a registry line and a drafted catalog, not a code change. The documentation is a separate mirrored build per language, and the six demo-video subtitle tracks are translated with everything else. **Where the translations come from, said plainly.** English is the source. The other ten were drafted by a local model running on our own hardware — nothing was sent to a third-party translation service — and **no native reviewer has signed any of them off**. Each is therefore recorded in the registry as unreviewed and tells the reader so once, at the foot of every page, in that language: the Filipino pages read *Isinalin ng makina. Ang Ingles ang masusunod.* — machine translated, English governs. A claims gate runs over every drafted string and writes nothing that would alter a claim, so a page is withheld rather than published wrong, and a figure, a framework name or a product name that a translation changed is a failure rather than a variant. A separate check compares each translated page's structural markers against the English page's, so an edit to an English page can no longer leave its translations quietly stale. **Honest status:** measured read-only on production on 2026-11-08 — the served display runtime carries all ten non-English languages, all ten marketing catalogs answer 200, and all ten translated documentation builds answer 200. Coverage for the most recently added language, Filipino: 3,169 of 3,169 marketing fragments across 28 routes, 3,191 of 3,194 portal strings, 212 of 212 partner-console strings, 42 of 42 documentation pages and 6 subtitle tracks, with the locale check reporting 0 failures. The endpoint agents, the browser extension and their installers are **English only** — they carry no translation mechanism yet — and so are the desktop status apps, which show the server's English level names. **Use case:** a bank in Jakarta or São Paulo evaluates the platform, buys it and runs it without a word of English in front of its staff. **Risk:** a security control nobody can read is a security control nobody uses, and a translation that quietly alters a compliance claim is worse than no translation at all.

83 · Payment path integrity — is the destination about to be paid the one that was attested? — `Roadmap` (Layer 1 built and merged 2026-10-26; Layer 2's credentialed read of the processor is not built, and nothing has run against a customer's settlement run) · Teams: SEC, RISK

This answers a different question from out-of-band verification (61). That control asks *did a human approve this payment*; this one asks *is the destination the money is about to reach the destination that was attested*. A shared classifier recognises the API calls of seven payment processors — Stripe, Adyen, PayPal, Square, Wise, Checkout.com and Modern Treasury — and sorts each one into **two money-moving operation classes that carry a verdict**: submitting a payment, and mutating a stored destination. A third class is named for a recognised call that moves no money (reading a balance, listing charges), so that "we looked and this was not a payment" stays distinguishable from "we did not recognise this host" — the same reporting discipline as everywhere else here. There are **eight divergence verdicts, of which exactly one clears a payment**; every other names what could not be established, including a payment seen at the processor that was never observed leaving (`payment_at_processor_with_no_observation`),

rather than being quietly treated as reconciled. **A matching processor token does not clear a payment, and that is the point.** On the common platform shape a payout carries only a reference to a stored account, and the call that redirects the money is a mutation of that stored account — after which every later payment looks identical to every earlier one. So a token match clears a payment only when the binding behind that token was itself attested and no unmatched destination mutation has been seen since. A product that compares the token rather than the destination passes the redirected payment. **What is not built, and a ceiling worth stating:** Layer 2, the credentialed read-back from the processor, is not built; and none of the seven processors returns a full destination (Stripe returns the last four digits and the routing number only), so even once Layer 2 exists, reconciliation reaches `consistent_but_partial` at best and never a full digit-for-digit match. **Honest status:** Layer 1 is merged and on `main` (2026-10-26), with the classifier vendored byte-identically into the RASP Python layer and the endpoint agent, and the intent registry, observations and declared-surface denominator in the policy service. Nothing has been exercised against a customer's live settlement run, the question of where a settlement account is legitimately established is unanswered, and the release gates in its own specification are unmet, so no higher tier is claimed. **Use case:** a finance system is told to pay a supplier; between the approval and the money leaving, the stored bank details are changed at the processor, and the change is caught because the destination no longer matches the one that was attested. **Risk:** the fraud that empties an account is rarely a forged payment — it is a redirected one.

Part 12 · Holding a privileged change (84)

84 · A privileged change is held until someone who did not make it approves — `Configurable` (built and merged 2026-11-13; three enforcement points, one of which has a true veto; not exercised against a customer's own privileged change) · Teams: SEC, RISK

The control the 2026 exchange breaches turned on, and it is not the payment: it is the change to the thing that AUTHORISES payments — a wallet control, a signing key, a service credential, a deploy target. A tenant declares the privileged object; a request is matched to it by **locator, never by name**, so nothing in the request gets to decide whether the request is privileged. A change to a declared object is then held until an approver who is **not the initiator** clears it. **One control, three enforcement points, and they are deliberately unequal.** The submission path is the only one with a true veto. The endpoint plane reports a change an endpoint observed, relayed by the control plane which has already authenticated the device, and it cannot stop that change — a limit of that point, reported rather than papered over. The deploy gate owns the `deploy_target` class, whose locator is a repository plus a ref plus an environment. The verdict vocabulary, the fail-mode comparison and the response shape live in one module and are imported by the others rather than restated, because a second copy of what "cleared" means is a thing that can drift into disagreeing about the only question this control exists to answer. The initiator is usually not a person, so three sources of identity are named in every response rather than assumed. **Honest status:** merged 2026-11-13, with the gate's own routes unpublished at the edge and asserted so by a test. Not exercised against a customer's real privileged change, and the endpoint plane's point has no veto by design. **Use case:** someone with production access changes the withdrawal address on a treasury wallet, or the signing key a release is built with; the change waits for a second named human instead of taking effect because the person who made it had the rights to make it. **Risk:** the breach that empties an exchange is rarely a stolen password — it is a quiet change to who is allowed to move the money.

Release validation update — 2026-11-22

Ninety-four changes merged between 2026-11-19 and 2026-11-22 were read one by one for this update. **No tier moved.** Most of them were fixes found by running the product on the two test machines, and several correct a limit this document had not stated. Dates are UTC. Each item below was read from the merged change and, where it says so, from the code or the published release.

One thing governs every item marked *not in a published installer*. The newest endpoint installer a customer can download is 1.0.19, published 2026-11-12. Version 1.0.20 was cut in source on 2026-11-21 and has not been

built, signed or published, and the release device gate has not been run against it.

- **The injection classifier on uploaded documents (6, 7, 11).** Measured 2026-11-19 on a Mac with the model pack shipped 2026-10-27: on uploaded documents the classifier marked benign logs, CSV files, code, Markdown, invoices and Chinese-language reports as injection so often that its score cannot tell a benign document from an attack. A finding from the classifier alone on an uploaded document, or on a file an AI coding agent reads (78), now warns instead of blocking until a retrained model passes a false-positive gate on each kind of document. Typed prompts are unchanged, and sensitive-data classes, malware, the pattern-based injection rules and the URL Trust Gate keep their full weight. Merged 2026-11-21, tested with a stand-in model only, not deployed and not in a published installer.
- **Images through the endpoint's own proxy (53, 54).** Until a fix merged 2026-11-21 that proxy did not read uploaded images: on a macOS test machine on 2026-11-19, 33 PNG uploads carrying a Social Security number, a card number or a cloud access key were allowed under a block-upload policy. They are now read with OCR and their text goes to policy, and an image nothing could read is blocked under fail-closed. One run with Apple Vision on a Mac blocked all three kinds; the Windows and Linux path is covered by in-process tests only. Not in a published installer.
- **Turning every redaction class on (8).** Measured on the development server on 2026-11-19: with a redact policy covering all 39 classes, an ordinary JSON chat request carrying a Social Security number came back with its closing punctuation swallowed into the redaction, so the request no longer parsed and was refused as a failed redaction. A fix merged 2026-11-20 trims each redaction to the value it covers; it is unit-tested without a model loaded, and whether it is deployed was not checked for this update.
- **One rulebook, one answer (14).** When several policies match, the lowest priority number decides and strictness only breaks a tie. On a Windows test machine on 2026-11-19 one endpoint gave two answers for the same Social Security number, because the coding-agent hooks and the clipboard helper ranked by strictness while the local proxy ranked by priority. The endpoint's evaluators, the Python SDK's local fallback, the Android policy core and the Java RASP now apply the engine's rule, held to one shared set of conflict cases in unit tests. Not run on a device.
- **A redact policy decides at every moment (54).** On the same Windows machine one prompt was redacted before the local models had loaded and blocked afterwards, under a tenant policy that says redact. From a fix merged 2026-11-20 the redact policy decides at both moments and a request that is harmful on its own stays blocked; on a test build there it was redacted at both moments, 64 times of 65 after a cold boot.
- **An unanswered question is not a refusal (47).** A prompt nobody answered within 60 seconds was recorded as the person refusing. It is now recorded as no answer given, still blocked, and asked again next time. Not in a published installer.
- **A coding-agent hook that cannot start (78).** On a Windows test machine on 2026-11-19, when the hook itself could not start, Claude Code allowed the prompt or tool call whatever the tenant had chosen, 30 times in one run. A hook that cannot start now follows the tenant's fail mode on Claude Code and

Codex, and an unreadable mode is treated as closed. Not run in a real session or on a test machine.

- **Captive portals (54).** The stand-down that lets a hotel or airport sign-in page load has run on macOS since 2026-07-21. Windows did not have it until a change merged 2026-11-21, which is unit-tested against a simulated registry and has not seen a Windows machine or a real sign-in page. Linux has none.
- **Ransomware response (39, 66) and the sensors (16).** The detectors have watched the whole of each user's profile since 2026-11-02, not the three folders this document names in entry 66. On the Windows test machine the sensor's record of who changed a file still covered those three folders, so 186 mass-deletion detections between 2026-11-09 and 2026-11-14 were all unattributed and none could be acted on; a change merged 2026-11-20 widens the record to the whole profile and has not run on any Windows host. On the macOS test machine on 2026-11-19 the sensor was registered, enabled and reported as passed while it had recorded nothing since 2026-11-05, so the response was refused as unattributed. From a fix merged 2026-11-21 the status check passes only when the sensor is registered and running; the updated sensor has been compiled and never signed or run. On 1.0.19, *passed* for this sensor means registered, not running.
- **Automatic quarantine (43).** A PowerShell script shipped in the Windows agent archive reached the quarantine verdict, because a short flag in the pattern list is the beginning of an ordinary one. Corrected by a fix merged 2026-11-20; not in a published installer.
- **Editor extensions (18).** Version 1.0.2 for VS Code, Cursor and Kiro. Until it, eight declared settings changed nothing and block mode wrote a detected secret to disk unchanged, measured 2026-11-19 in VS Code 1.144.0 and Cursor 2.1.26 on a Mac. Every setting is now read by what it controls, and under block a save follows the tenant's fail mode. In VS Code the acceptance run read 150 checks passed, 0 failed and 45 not checked of 195. Not run: a save in Cursor or Kiro, Windows, Linux. The Visual Studio edition has never been compiled.
- **Runtime protection in Java (21).** Until a change merged 2026-11-22 the Java RASP did not evaluate a tenant's policies, so a tenant's block policy never blocked a call made from a Java process; its built-in detectors ran as configured. It now evaluates each synced policy and applies the highest-priority match, and a policy that depends on something the process cannot see is reported as not evaluated. 119 unit tests pass on one Mac. No Java application has been instrumented end to end.
- **Client SDKs (59).** Read from the services' route tables on 2026-11-19, the Python and .NET SDKs asked for a policy verdict at an address no service serves. Both now ask the control plane's published route, tested against stand-in responses only. The other seven ask at routes the hosted application address does not publish, and receive a server verdict only where the policy service itself can be reached.
- **The browser's link to the endpoint on Windows (17, 73).** It did not work on a stock Windows install until two fixes merged 2026-11-19 and 2026-11-20: registration was refused, and once registered the endpoint could not tell which extension had called it. Neither fix is in a published installer. The extension's source version is 2.3.28, not store-published.
- **Android (68).** Until a fix merged 2026-11-20, blocks left the device only when someone opened the app: on a tablet on 2026-11-20 the portal showed 0

blocks while the device had refused 272 lookups. Blocks now travel on the regular check-in; not yet in a released build. Measured on an emulator on 2026-11-19, a browser set to a named secure DNS provider, or an app that asks a public resolver directly, is not seen by the filter.
- **Status indicator (58), and the value view.** The menu bar and tray icon is the company's shield and carries a second mark for the AI off switch; the status apps can mark a rise in a value figure with a sound, muted by default, and an animation that the organisation or the person can turn off. Rendered on a Mac and a Windows 11 virtual machine; not in a published installer.

Status-page rows that have no numbered entry here were updated in the same change: the AI off switch on the phones (still `Roadmap`: the card now says AI is off only while a live filter is refusing, checked on an emulator and a simulator and not on a phone, and refusal by connection is still not built), the AI off switch for a fleet, and the value view. Four rows are new, all at `Roadmap`, which means not available:

- **AI off by voice.** The rule and a listener are built; no microphone code, speech engine or voice training is in the product.
- **CyberArmor.AI Messenger.** Protocol libraries and a screen preview on sample data; nothing sends a message, there is no server, and none of it has been security-audited.
- **CyberArmor.AI Meet.** A booking service, booking pages and an app preview, all unit-tested; the production deployment does not start the service, and real audio, video and calendar linking are not built.
- **CyberArmor.AI Red Team.** A specification of the consent, scope and ownership checks that must exist before any testing does. Nothing is built.

As on 2026-11-19 these are deliberately not numbered entries here, because this document's count is a published figure on several pages.

Release validation update — 2026-11-19

The two mobile endpoint entries (67, 68) re-measured for the site update. Both counters were read from each region's control plane, not from a device's own report of itself, and **no tier moved**: both stay `Configurable`, matching the status page.

- **iOS and iPadOS (67).** The US control plane holds eight registrations, the newest on iOS 27.2.1 (agent 1.0 (27)) reporting 6,227 AI lookups observed and 72 blocks in its last 24 hours. Singapore holds a ninth, on iOS 27.2.1 (agent 1.0 (26)), reporting 4,951 lookups and 48 blocks.
- **Android (68).** Singapore holds two Android 16 registrations, the newer on agent 1.0.2 reporting 5,226 AI lookups observed and 4,807 blocks in total; the US control plane holds a third, on agent 0.1.0, reporting 2,160 lookups and 2,112 blocks.
- **Not re-measured, and therefore unchanged:** the Safari web extension's own counters (no new Safari reading was taken, so the 2026-10-07 sentence stands) and the iOS App Proxy (69), which still has no counter in any registration record because it starts only under an MDM per-app VPN profile.

Four capabilities new to the status page on the same day — the AI off switch on computers, on the surfaces beside the agent, on the phones and the watch, and for a fleet — plus protection levels, the value-and-friction view and the AIProtect.app front door, are rows on `marketing/app/status/page.tsx` and in `docs/architecture/capability-status.md`, and are **deliberately not numbered entries here yet**: this document's count is a published figure on several pages, and moving it is a change of its own.

Release validation update — 2026-11-14

Five changes merged on 2026-11-10 and 2026-11-13 that move a line on this page. Each was read out of the merged code rather than from its pull request's summary.

- **A `block_upload` policy now applies to a desktop AI client (54, 11).** This is the one with a customer consequence: before #957 the endpoint proxy asked policy about every request without saying what kind of request it was, so every upload policy was skipped and a file the browser extension would have stopped left the ChatGPT desktop app, Claude Code or Codex. The proxy now labels the request and carries the extension's own six content fields, so one rulebook covers both.
- **A flag is visible, and every count has a list behind it (58).** #959: a `warn` dispatches a desktop banner, and a durable 200-entry record of blocks, flags and applied redactions sits beside the health file with ten fields each and lifetime totals that survive a restart. Stated precisely, because the distinction is the point: the banner is **dispatched, not delivered** — nothing on that path can ask whether it was displayed, suppressed or denied — so the capability is **flagged and let through**, never "the user was warned".
- **An app that distrusts the proxy's CA is named and told to restart (55).** #960: attributed to a process from its port, told once, listed with the CA file's SHA-256 and modification time, cleared when that process handshakes. Unattributable clients are counted, not dropped.
- **The portal stops undercounting its own proxies (38).** #958: the AI-traffic-proxy card read "Not installed, 0 reporting of 0 installed" on a fleet where every agent ran one, because the page never read the fields the agents sent.
- **A privileged change waits for a second person (84).** #938, a new entry: a change to a wallet control, signing key, service credential or deploy target is held until an approver who is not the initiator clears it, matched by locator rather than by name, across three enforcement points of which only the submission path has a true veto.

Not given a row: the cawin driving harness (#956) is internal tooling for recording the demo VMs, not a customer capability.

Release validation update — 2026-11-08

Measured 2026-11-08 from this laptop, read-only: the production and Singapore hosts' running containers and their served assets. Production's checkout is on `main` at `0db63297a`, and its control-plane, marketing and documentation containers were recreated from it the same day (19:18–19:20 UTC); Singapore's control-plane image was recreated at 19:20 UTC and carries the same `protection_levels.py`. What changed for a customer since the 2026-11-04 edition of this page:

- **Secure by default, in one step, and measured (81).** A new capability, and the shortest path there has ever been from buying the product to being protected by it: five protection levels, applied in one step, with the posture *measured* back out of the settings rather than reported from what was stored. The hosted control plane and the customer portal are deployed on production and Singapore. The matching per-device view — desktop panels and trays, the Linux status command, the browser extension, the Office and IDE add-ins, iPhone, Apple Watch and Android, the ROS 2 robot and the endpoint's own proxy — is built and tested on every one of those surfaces and ships in endpoint agent 1.0.18 and browser extension 2.3.25; none of it has been read on a customer device yet.

- **Eleven display languages (82).** A new capability. English, Simplified Chinese, Brazilian Portuguese, Latin American Spanish, Korean, Japanese, Thai, Vietnamese, Indonesian, Malay and Filipino, across the marketing site, both portals and this documentation, with translated video subtitles. All ten non-English languages are live and every one is machine-drafted, unreviewed, and says so at the foot of each page. The agents, the extension and the installers remain English only.
- **Payment path integrity reaches this page for the first time (83).** A new entry, at `Roadmap`. It was built and merged on 2026-10-26 and no status surface showed it: a customer reading either public page would not have known the work existed. It is published at the tier its evidence supports — the comparison in the submitting process is built, the credentialed read-back from the processor is not, and nothing has run against a customer's live settlement run.
- **Endpoint agent 1.0.18 drops Intel Macs and ends a security waiver (16).** Every macOS package from 1.0.13-rc.15 through 1.0.17 installed `cryptography` 48.0.1, which carries three live advisories (PYSEC-2026-3552, -3553 and -3554: a PKCS#7 Bleichenbacher oracle, exponential X.509 path building, and wildcard DNS names escaping their permitted subtrees), under a waiver re-approved at each release. The cause was that one package served both kinds of Mac: re-measured on 2026-11-07 against the package index, all 17 macOS wheels published across the four fixed `cryptography` releases are Apple-silicon only, so no single lock could both fix the advisories and run on an Intel Mac. 1.0.18 therefore requires an Apple-silicon Mac, ships `cryptography` 50.0.0 or later, and the waiver is deleted rather than renewed. **Customers on Intel Macs stay on 1.0.17 and keep those three advisories**; the Windows and Linux packages are unaffected.
- **A browser enrolled from its own options page is no longer called protected before it has a policy (17).** Browser extension 2.3.25. Enrolling from the options page wrote the credential but started no policy sync, and the popup and options page read as protected anyway — for up to ten minutes, which is what the extension's own sync cadence is written to do. Enrolment now syncs at once, arms the sync alarm, and says it is not protected until that first sync lands.
- **One price, and one plan.** $19.99 per user per month, or $199.90 per user per year, with up to three enrolments per user and the fourth refused; individuals, families, teams and businesses all buy the same plan. This replaced the four-tier catalogue on 2026-11-07. Read on the live pricing page 2026-11-08.
- **The service-side decision point is named CyberArmor.AI Trust Gateway (19)**, in English and Simplified Chinese, on Onboarding & Downloads — so the hosted gateway and the endpoint's own proxy are no longer both called "the proxy" in the surfaces a customer reads.
- **A Log in button that routes through the region chooser**, live since 2026-11-06: a customer picks their region and whether they are a customer or a partner, the choice is remembered, and it is never inferred from where they appear to be connecting from. The portals link back.
- **The portal stops running out of database connections under telemetry load.** The hot telemetry reads are indexed, telemetry ingest has its own connection pool, and one worker (not eight racing) builds those indexes, after eight racing workers deadlocked and left invalid indexes behind. This is why an administrator's page load could report the service as down while nothing was actually wrong with it.
- **The published fleet ceiling now accounts for telemetry history.** The per-server endpoint ceiling falls as stored telemetry grows — something the original load tests could not see, because they ran against an empty history. Anyone sizing a deployment should read the capacity guide rather than the older headline figure.
- **The CJK prompt-injection model's failed promotion gate is now recorded in the repository (6).** The Chinese-and-English model that decides Chinese, Japanese and Korean text did not pass its own promotion gate and is deployed on that narrowed route by an explicit decision. That was true before this edition and nothing in the repository said so; the gate record now does, and this page already treated the model as not independently validated.

Release validation update — 2026-10-28

Measured 2026-10-28 from this laptop, read-only: each host's checkout and its running containers. Dev, production and Singapore are at `c9b913e7d`, and the trust gate, AI router, control plane, compliance and detection containers on all three were recreated from it that morning. Endpoint changes are on `main` and ship in endpoint agent 1.0.13-rc.15, which is not yet published; none has run on a customer endpoint. What each change gives a customer, and what they see:

- **Card-and-PIN phishing is stopped before the PIN is typed (1).** Live on the hosted gate. A page that asks for a card PIN is blocked, a card, CVV, OTP or CNIC form counts as a credential form, and 13 Pakistani banks are protected brands. A bank's staff, and AI agents working for them, cannot be walked into a look-alike page; the user sees a block page that says the page looks like phishing. Measured on the engine, not yet in a browser against the deployed gate.
- **Audio and video leave only after they are read (11, 53, 71, 74).** Video frames are read on the server, and on a Mac on the device; from rc.15 audio is transcribed on the device, so a card number spoken into a voice note or a recorded call is found before an AI service receives it. A recording that used to be held is decided by policy instead, with no file sent to the server for transcription. The spoken track of a video is not transcribed.
- **AI coding agents read media the way the browser does (78).** From rc.15 a screen recording, screenshot or scanned PDF given to Claude Code or Codex is checked before the agent reads it, and the bundled safe-fetch works as intended. Engineering teams can use coding agents on incident material without customer data leaving through the agent. Measured in-process on a Mac only.
- **Ransomware is stopped on Windows, and the user is told (39, 66).** The kill ran on a Windows 11 test VM with the rc.15 fixes applied by hand: the simulator was stopped after 7 of 40 (and, on a second build, 6 of 40) demo files, and the portal recorded the kill. From rc.15 the person at the keyboard sees one banner per incident saying whether the program was stopped, and the status app gains a Show notifications switch (58). The Windows sensor that makes this possible moved from Roadmap to Configurable after its first run on a Windows host.
- **The portal says one thing (26, 28, 29, 57, 80).** Every view counts a block the same way and opens on the last 24 hours; Identity / SSO and Settings are separate pages; the compliance page shows each framework's last result and when it ran; SIEM connectors carry their vendors' names. On the Singapore region's portal: Run all assessed 45 of 45 frameworks, an attestation with a policy document moved NIST CSF from 33% to 39% (the Attested count stayed at 0 and the row did not name the file, both not yet fixed), and the whole-trail audit check found no forks over 5,131 records.
- **Smaller changes, from rc.15:** a link or IP address on its own no longer counts as personal data (8; also in the hosted AI router), the Windows clipboard row in the tray measures the helper (79), and the agent's mitmproxy and its dependencies are updated (mitmproxy 12.3.0, Tornado 6.5.10, cryptography 50.0.2).

Release validation update — 2026-10-23

Measured 2026-10-23 from this laptop, read-only: production's checkout and running images, the audit service's own health document, and `gh release list`. Dev and Singapore were not measured for this update.

- Production's checkout is at `262f597` (merge of #625). `pull_and_rebuild_what_changed.sh --no-pull --dry-run` compared the 22 running service containers against it and found nothing to rebuild; the docs and marketing containers could not be compared.
- Audit ledger (26, 32): `checkpoint_signing` reports ML-DSA-87, enabled, worker running, 0 chains failing.
- Latest published pre-release: endpoint agent 1.0.13-rc.11 with browser extension 2.3.15-rc.11 (2026-10-22). The 1.0.13-rc.12 release is merged on `main` and not yet published.
- New entries 77–80 are at the tier each states; none is measured on a customer endpoint.

Changed inside existing entries since 2026-10-13 (each at the tier its entry states):

- Prompt-injection detection (6): Chinese, Japanese and Korean text is decided by a Chinese-and-English model; every other language by the existing model, unchanged.
- Redaction (8): 39 classes, with China's resident ID and mobile number and a debit-card PIN typed next to a card number; pasted secret assignments (`…_SECRET=` , `AWS_SECRET_ACCESS_KEY=` , `AUTH_TOKEN=` , a database URL's password and more provider key formats) are secrets on every user-data surface, and a placeholder is not.
- Reputation (1): known-bad lists in the signed snapshot, refreshed hourly, guarded so that no line can block a ranked domain, an AI service or a platform.
- Local-first decisions (73): the device checks the page the user already has open (on by default when the agent has its models), and the server crawls only what it does not clear; local models are on by default through a staged rollout on Apple-silicon Macs and Windows.

- Status indicator (58): a time-boxed Pause of AI-traffic inspection is available by default on macOS and, built and not validated on hardware, on Windows; organisations turn it off by device management or tenant setting.
- Process termination (66): ransomware response kills the process the OS sensor saw, never a guessed one; built, not validated on hardware.
- Accounts and tenants (35, 36): users are invited by email, a tenant can have a device limit, and accounts, partners, tenants and devices can be disabled, retired, suspended or terminated under one scoped, stepped-up and audited lifecycle; a partner member who belongs to several partners picks one after sign-in and passes that partner's MFA.
- A Simplified Chinese interface for the portal, partner console, marketing site and documentation is built and in native-speaker review; it is not offered to customers yet.

Release validation update — 2026-10-13

Measured 2026-10-13 on the dev, production and Singapore hosts and on the GitHub release; a green build does not promote a feature.

- All three hosts were deployed at the same commit and every service image matched it (the deploy script compares the running containers, not git).
- Endpoint agent 1.0.12 beta: macOS PKG/DMG and Windows MSI/offline ZIP are on all three hosts' download paths and on the `v1.0.12-beta` pre-release with their SHA-256. The macOS artifacts are Developer ID signed, notarized and stapled; the Windows MSI and ZIP are Authenticode signed. On a Windows 11 VM the MSI installed over 1.0.11, uninstalled and reinstalled, and the local engine answered its health check. The demo rehearsal on the VMs was skipped for this release. Not yet installed on a customer endpoint.
- Browser extension 2.3.11 (Chromium/Edge, Firefox, Safari): source zips with SHA-256 on the pre-release. Browser-store publication is a separate step.
- Android (68): the 1.0.0 APK is on all three hosts' download paths, offered on the customer portal's Onboarding & Downloads page with its SHA-256. The device row stays at its tier.
- iOS (67): build 17 is not recorded here as on TestFlight; that needs a check in App Store Connect.
- New entries 73–76 are `Configurable`: built, released and deployed, not yet measured on a customer device or robot.

Release validation update — 2026-10-09

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.

New since the 2026-10-03 update

- Endpoint agent 1.0.11: macOS PKG/DMG served from production since 2026-10-11 (release `v1.0.11-beta`). Both are Developer ID signed, notarized and stapled, and Gatekeeper accepts them (checked 2026-10-11 on the artifacts; the served bytes' SHA-256 was checked against the release on the production host). Not yet installed on an enrolled Mac.

- Windows stays on 1.0.10 (MSI/offline ZIP, served since 2026-10-05): the 1.0.11 Windows build did not run, because its signing workflow's hosted Windows job was refused by GitHub billing. The Windows 1.0.10 signature has not been re-checked for this update; the 1.0.7 checks below are the most recent.
- ROS 2 robotics agent (70): re-validated on robot hardware 2026-10-07 on a Raspberry Pi 5. Latching e-stop produced zero downstream output, a 9.0 m/s command was delivered to the motor driver as the 2.0 m/s policy limit, the latch survived a restart and was released only by a signed token, and telemetry reached the control plane.
- iOS (67): live counters read on a physical device 2026-10-07. DNS filter enforcing and logging since 2026-09-24; Safari extension counters (pages checked, phishing blocked, data redacted, warnings shown) confirmed on the same device; enrolled across multiple production tenants. Still TestFlight.
- iOS App Proxy (69): wired into the app lifecycle 2026-10-07, so the provider now starts and reports its coverage state. Running it on a device under an MDM trust profile is the remaining step before promotion.
- Android (68): APK running on physical hardware in multiple production tenants, with audit records confirming device connection and counters; the Firefox extension has run on the same device in production since 2026-09-27.
- Endpoint document scanning (71): since 2026-10-06 text, DOCX, XLSX and PPTX uploads are scanned by the local engine, so a busy detection service no longer turns into a blocked upload; PDF and image uploads still go to the server-side OCR extractor (53). The browser extension also routes typed prompts through the local engine.
- Voice-triggered robot stop (72): signed multilingual phrase set and bench harness built; openWakeWord selected and wired into the harness 2026-10-09, with 29 of 29 detector tests passing through openWakeWord's real feature pipeline. No per-language stop-phrase model is trained yet.

Carried forward from 2026-10-03

- Signed macOS 1.0.7 PKG/DMG: notarization, stapling, Gatekeeper and served-byte hash verification passed. Automatic agent binary upgrade is not implemented.
- Signed Windows 1.0.7 MSI/offline ZIP: signature, timestamp and revocation checks passed and Windows 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. From release 1.0.13-rc.10 on by default through a staged rollout, on Macs with Apple silicon (macOS 14 or later) and Windows only; a tenant administrator can turn them off. Not yet run on a customer fleet.
- 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 extensions: running on devices, Safari on iOS since 2026-09-24 (device counters read 2026-10-07) and Firefox on a production-enrolled Android tablet since 2026-09-27. 2.3.7 passed

synchronized source and package checks; source is now at 2.3.10.
Browser-store publication is a separate release step.