BUYER'S GUIDE

The 2026 Buyer's Guide to AI Pentesting for Medical Devices

How to evaluate what produces evidence a regulator will accept

Whitepaper · ELTON Cyber · September 2026

WHITEPAPER

KEY TAKEAWAYS

- AI made vulnerability discovery cheap. Validation, prioritization, and proof of remediation are now the scarce work, and they are what a medical device manufacturer is actually buying.
- A point-in-time pentest was designed to produce a submission artifact. FDA's postmarket expectation is a continuous program, and the two are priced, scoped, and evidenced differently.
- A real AI-driven pipeline is a harness around a model, with the test plan derived from the device before the first packet is sent. The model is a swappable component, so judge the harness.
- The most capable cyber models of 2026 are reserved for verified defenders, and most manufacturers will never complete that vetting. Who holds the access, and under what controls, is a procurement question.
- Building a pipeline internally means building and maintaining a software product. A prompt that works on Tuesday is the easy part.
- Discovery without management is a liability. Require the platform that verifies exploitability on the device, dismisses the rest with evidence, and proves the fix.

Executive summary

Most medical device penetration testing was bought to satisfy a submission. A scoped engagement, a report, a set of deficiencies answered, and a cleared device. The model worked when a device shipped, was patched rarely, and faced attackers who needed weeks to find a bug.

In 2026 the gap between being tested and being defended is wider than it has ever been. AI-driven discovery finds candidate vulnerabilities at machine speed, in binaries, without source. Regulators expect a program that runs through the device's supported life. And the models that do the most capable discovery work are gated behind vetting programs that a manufacturer cannot complete on the side.

Almost every cleared device was tested. Whether that testing produced evidence a reviewer will accept, kept up with the release cadence, and told engineering which of hundreds of findings need a fix this sprint is a separate matter.

Buyers face a harder decision than they did three years ago. The market now includes consultant-led engagements, scanner suites, and continuous pipelines that use frontier models to build their own tools. Each has strengths and each carries a trade-off that procurement rarely surfaces.

This guide is written for:

- Product security and cybersecurity leads at device manufacturers who own the testing budget
- Regulatory affairs teams who have to defend the evidence in a 510(k), De Novo, or PMA
- CISOs and engineering leaders who must absorb AI-scale vulnerability volume without adding headcount
- Anyone who has been asked whether the company should build its own AI-driven pipeline

Inside, we reset the evaluation criteria, walk through the three testing models buyers actually encounter, explain what a real AI pipeline is and why the model inside it is interchangeable, and make the case that the purchase is two things: discovery and the management platform that turns discovery into proof.

Why the way manufacturers buy testing is broken

The purpose of testing a medical device is to reduce the chance that a reachable weakness becomes a patient safety event or a recall. A testing program that does not change that likelihood, with evidence, is not doing its job.

How once per submission became good enough

Premarket cybersecurity testing became a fixed event because the submission is a fixed event. A manufacturer scopes an engagement in the months before filing, the consultants test, the report goes into the eSTAR, and the team answers deficiencies. Time-bound engagements made scope limits normal, and reviewers who needed a document to read turned the report into the deliverable.

Those constraints hardened into industry practice. Attackers do not operate on submission cycles, and neither does FDA's postmarket expectation.

Scope chosen before the first packet

A consulting engagement is tester-led. A human with only days to work chooses the subset of the device that in their judgment matters most. In our experience across 1,000+ devices, a typical engagement covers 5 of the 24 testable areas on a connected device, chosen before the first packet is sent. A wrong guess is silent, because the engagement ends on schedule either way.

The question that goes unasked is simple: what part of this device has never been tested?

Findings volume tells you almost nothing

A thick report reads as thorough. A high count of low-impact findings does not lower risk if none of them are reachable from the device's real attack surface. One exploitable service-port credential in the technician menu matters more than forty theoretical library CVEs behind a TLS boundary that blocks the path.

Risk falls only when a condition an attacker can use on the deployed device is removed, and a findings count removes nothing.

The report that leaves with the consultant

Remediation guidance in a legacy report is generic by construction, on the order of "sanitize input" or "use stronger encryption," with no file, line, patch, or verification attached. Engineering interprets the advice, ships a fix, and nobody checks it until the next engagement, which is a year away and starts from zero.

The findings that never make the report are the larger problem. Writing up a finding takes hours the engagement ran out of. A median consulting pentest runs about $100,000 and returns fewer than ten true positives, and the testers usually know of more than they wrote up.

The obligation the engagement model ignores

FDA's premarket guidance is explicit about cadence. Under the Secure Product Development Framework, "cybersecurity testing should occur throughout the SPDF," and "after release, cybersecurity testing should be performed at regular intervals commensurate with risk (e.g., annually)." Under the Total Product Life Cycle expectation, manufacturers must "ensure they have appropriate resources to identify, assess, and mitigate cybersecurity vulnerabilities as they are identified throughout the supported device lifecycle." Section 524B made the postmarket plan a condition of clearance.

A once-per-submission engagement produces one artifact for one date, which cannot satisfy a continuous obligation.

Why continuous testing wins, and what a real subscription includes

Continuous testing changes the economics of the same work. The reasons are structural, and each one shows up on the invoice or in the submission.

Findings stay development artifacts

When testing runs as code lands, a finding is caught in a sprint and fixed before a submission exists, staying a development artifact under the manufacturer's control. Under a model that tests once before filing, the same finding surfaces inside a regulatory document and has to be answered there. Continuous testing reduces both the number and the severity of findings that appear in formal review.

Stop paying to start over

A consulting engagement restarts from zero each time, with fresh scoping and reconnaissance on a fresh invoice, so a product year with six releases means six restarts.

A subscription maintains traceable test cases and a digital twin of what was covered. When release six lands, the platform diffs it against release five and retests what changed. In a recent example, 847 test cases carried forward, 2 of 5 components had changed, and the Bluetooth stack, DICOM listener, and web UI bundle inherited their prior evidence because nothing in them moved. Six releases become one history with delta testing.

Depth compounds

Coverage resets every year in a point-in-time model and accumulates in a continuous one: one engagement reaches about 7 of 40 testable areas on a typical device, while two years of continuous testing reach about 34 of 40, because each release targets what has not been tested or was never a focus.

Capacity is reserved, and speed follows

An annual subscription blocks capacity before the release calendar exists, which removes the scheduling conflict when a date slips, the expedite fee, and the six-week lead time. Milestone testing closes in under two days, and a report can be regenerated against the current CVE set without a new engagement.

Six things a subscription must include to be one

Vendors have started calling retainers "subscriptions." A subscription in the sense that matters provides:

- → Test case traceability across engagements, so scope is inherited rather than rewritten
- → A product model that shows which components changed since the last test
- → Every finding classified as directly exploitable, conditionally exploitable, or a weakness, with the reasoning attached
- → Fixes written to the file and line, verified before delivery
- → Reports that regenerate against the current CVE set without a new statement of work
- → Transparency on how AI is used, and how that efficiency reaches pricing and coverage

If a vendor cannot show these six, the word on the contract is doing the work.

The three testing models manufacturers actually encounter

Testing approaches differ in design and in what they were built to produce. In 2026 a buyer will meet three, and each carries a trade-off that is not obvious during procurement.

1. Consultant-led penetration tests

The familiar model. Skilled testers assess a defined scope over a fixed period, with judgment shaped by experience on similar devices.

STRENGTHS

- → Deep expertise and creative attack paths
- → Context-rich interpretation of what a finding means clinically
- → A report format reviewers recognize

CONSTRAINTS

- → Scope fixed before the first packet, typically 5 of 24 areas
- → About six weeks from kickoff to findings
- → Generic remediation, no fix verification, no carry-forward

The trade-off is breadth and continuity. A time-bound engagement produces a high-quality snapshot of a limited slice of the device, and the snapshot ages the day it is delivered.

2. Scanners and static tooling

To get coverage and repeatability, many manufacturers assemble SAST, SCA, SBOM CVE matching, and fuzzers into a suite that runs in CI.

STRENGTHS

- → Repeatable, and it runs on every build
- → Broad in the dimension it measures
- → Low marginal cost per run

CONSTRAINTS

- → No reachability: a CVE in the SBOM is a version match with no proof of an exploitable path
- → Cannot exercise hardware interfaces or the device as deployed
- → Volume without context, so triage moves to engineering

The trade-off is meaning. A scanner reports that a vulnerable version is present, but whether the vulnerable function is reachable from any entry vector on the device, the question a reviewer and an attacker care about, is beyond it.

3. Continuous, AI-driven pipelines

The newer model combines a persistent model of the device with an orchestrated toolchain that a frontier model helps build, running on code, firmware, and real hardware every release.

STRENGTHS

- → Coverage as a written plan derived from the device, executed every release
- → Discovery at machine speed across firmware, protocols, and hardware interfaces
- → Findings carry forward and evidence accumulates release over release

CONSTRAINTS

- → Requires a management platform to absorb the output
- → Requires organizational readiness to receive more true positives than before
- → Quality depends entirely on the harness around the model, which is invisible in a demo

The trade-off is the harness. Two vendors can name the same model and produce unrecognizably different evidence, because the model is the smallest part of the system. The rest of this guide is mostly about how to see the harness.

A note on autonomous web-app pentesters

A class of tools built for enterprise web applications now markets itself as autonomous pentesting. They are strong at what they were designed for: exploring a reachable HTTP surface at machine speed and producing a working exploit. A medical device is a different shape of problem. The surface is firmware, hardware interfaces, clinical protocols like DICOM and HL7, and a service technician's USB port, and the customer is a regulator who asks for coverage and traceability rather than a findings list. Evaluate them for what they are, and do not let a web-app benchmark stand in for device coverage.

What a real pipeline is, and why the model is a component

"A pipeline, not a prompt" is the shortest description of the difference between testing and asking. Pointing a model at a product and asking it to find vulnerabilities is not testing. It is a prompt, and anyone can write one.

The chain a reviewer can walk

A threat-led pipeline builds the test plan before the test runs and derives it from the device itself. In order:

1. **Digital twin.** A model of the release's security architecture, built from the manufacturer's own quality-system artifacts: architecture and data-flow diagrams, the SBOM, interface specifications, documented countermeasures.
2. **Threat analysis.** Every twin component is mapped to the threats that apply to it, on the MITRE EMB3D backbone with medical device extensions.
3. **Test case generation.** One or more test cases per applicable threat, each with explicit pass and fail criteria, and tooling built to order.
4. **Execution.** Test cases run on code, on firmware, and on the physical device, remotely through a test appliance on the manufacturer's bench.
5. **Verification and classification.** Each result lands in an exploitability state, is rated on the FDA-qualified MDDT rubric, chained through a dependency graph, and carried into a VEX statement with evidence attached.

Coverage is the threat register. The test plan is that register with a test case filled in on each row, and once results are filled in it doubles as the coverage record. A reviewer can walk the chain in either direction: twin, property, threat, test case, result, finding, disposition, VEX.

The model's two jobs and the boundary around them

The model has two jobs in this design. It infers which threats apply to which components, and it writes the tools and test logic that exercise each threat. Five agentic loops run around the clock against protocol tooling, firmware tooling, hardware interfaces, application surfaces, and exploit validation, each one finding gaps in the toolchain and writing what is missing. The toolchain now holds thousands of tools that know how to test a medical device.

Whether a finding is real is decided by deterministic execution on the code, the firmware, or the device, never by the model. And the tool-writing model never sees the customer. Source code, firmware images, architecture documents, findings, and device identifiers stay inside the tenant, where in ELTON's design they are processed only by open-weight models on ELTON compute with no outbound calls. What goes to a tool-writing model is the vendor's own engineering work, such as tool specifications, documented protocol behavior, and harness code. We ask a model to write a tool. We never ask it about you.

A human gate sits in front of every result, so nothing the pipeline surfaces reaches a customer unreviewed.

Models are interchangeable, on purpose

When the model's job is bounded this way, swapping it is a configuration change that leaves the test plan, the coverage record, and the verification levels untouched. The one thing that moves is the ceiling on the reasoning layer, meaning how well the model reads a large software system, the quality of tool it can write for binary work without source, and its refusal rate on the dual-use edge where exploit validation lives.

Interchangeability matters for a reason beyond flexibility. Models turn over every quarter, access terms move without much notice, and an export control can remove a model outright, as on June 13, 2026, when Fable 5 and Mythos 5 were disabled for every customer under a US directive and stayed dark for eighteen days. A pipeline that depends on one model is a pipeline with a single point of failure that the vendor does not control.

Model access is now a procurement question

The strongest discovery work in 2026 comes from models most manufacturers cannot obtain. Whether a vendor holds that access, and how, belongs on the evaluation sheet.

The gates in 2026

Anthropic's Claude Fable 5.1 and Claude Mythos 5.1 are the same model with different safeguards. Fable 5.1 is generally available and restricts penetration testing, exploit generation, and binary vulnerability scanning. Mythos 5.1 permits that work for vetted US organizations under the Cyber Verification Program, which grew out of the invitation-only Project Glasswing preview that seeded Apple, AWS, Microsoft, CrowdStrike, and more than 40 other organizations.

OpenAI's Trusted Access for Cyber, branded Daybreak, runs in two tiers. Daybreak Blue provides GPT-5.6 Sol to verified defenders for triage, code review, malware analysis, and patch validation. The Red tier adds GPT-5.6 Cyber, which is built for authorized penetration testing, red teaming, exploit validation, and controlled vulnerability research. Red requires additional approval and, per OpenAI's published requirements, SOC 2 Type II or ISO 27001, single sign-on, MFA, role-based access, usage logging, incident response documentation, company-controlled devices, and hardware security keys for each approved individual. A company that provides these capabilities to customers goes through a separate review, the Daybreak Cyber Partner Program.

On OpenAI's internal benchmark of exploit-chain, authentication-bypass, and privilege-escalation tasks, GPT-5.6 Cyber completes 95 percent, against 57.3 percent for GPT-5.5 Cyber and 1.5 percent for the standard model. Independent evaluation by the UK AI Security Institute found Claude Mythos Preview a step up over prior frontier models on capture-the-flag and multi-step attack simulations. A serious adversary would want exactly this capability, so a testing program should be putting it to work on your device before an adversary does.

Three lanes for the tool-writing model

When AI Vulnerability Testing is elected, the manufacturer chooses which model writes the tools, and a good vendor supports all three lanes.

- **Frontier, through the vendor's access.** The vendor holds the approvals, runs the models under the controls both companies require, and delivers evidence. The manufacturer neither touches the model nor has to pass the vetting. ELTON operates this way: OpenAI evaluated the ELTON service and granted Trusted Access for Cyber to protect medical devices, and ELTON runs Anthropic Claude Mythos through verified-defender access. Customers use these models through ELTON and do not obtain them from Anthropic or OpenAI. Approval under these programs rests on operating controls and a demonstrated history of authorized work, and ELTON's history is 13 years and 1,000+ devices.
- **Open-weight, hosted locally.** Open-source weights on the vendor's compute inside the manufacturer's tenant, with no vendor account, no telemetry, and no outbound model calls, so no frontier vendor is in the loop at all. The trade is a lower reasoning ceiling for the shortest possible data path.
- **Bring your own model.** A manufacturer that already licenses a model, or holds its own Trusted Access or verified-defender approval, points the pipeline at its own endpoint. The harness, the tools, and the tradecraft stay the vendor's.

The data boundary should be the same on every lane. A tool-writing model receives only the vendor's engineering inputs, meaning tool specifications, documented protocol behavior, and harness code, and returns tool selection, discovery code, and exploit code. Customer documentation, code, firmware, findings, and device identifiers are never sent to it. In ELTON's design, customer data inside the tenant is processed only by open-weight models on ELTON compute. All three lanes produce the same coverage record, verification, and evidence, and differ in reasoning ceiling, in who is in the loop, and in cost.

What to require of whoever holds the credential

- Which programs the vendor is approved under, in writing, and for which use cases
- The controls on the credential: who can invoke the model, from what devices, with what logging and human oversight
- The data boundary: an exact statement, in one sentence, of everything that enters a model
- The swap mechanics: what happens to your program if a model is withdrawn, re-gated, or replaced
- Whether you can choose the model class, and what changes in the deliverable when you do

Why building it yourself means building a software product

Every manufacturer with a capable engineering team will consider building this in house. The temptation is rational. A frontier model with a shell and a target finds things, surprisingly often, on the first afternoon.

The temptation

Prompting is easy. A security engineer with model access can point it at a firmware image and get plausible, sometimes real, findings in an hour. The demo writes itself, and it is the wrong evidence for the decision, because the demo answers whether a model can find a bug and the program has to answer whether the company can run testing that is controlled, evidenced, and consistent across every release for the life of the device.

What controlled, evidenced, and consistent costs

A pipeline that a reviewer can trust needs, at minimum:

- → A sandbox with an egress policy, so an agent cannot reach anything it was not pointed at, and a record when it tries
- → A credential broker, so no secret lives in a prompt or a log
- → Tooling that drives USB, Bluetooth, JTAG, UART, and proprietary radios against a real device rather than a simulation of one
- → Orchestration that keeps an agent on the test plan, with stop conditions and an action log per step
- → Deduplication and correlation across runs, because two runs of the same model on the same target produce two different findings lists
- → A verification harness that proves or dismisses every candidate on the device, and stores the evidence
- → A human review gate staffed by people who have tested this class of device before
- → The compliance controls the model vendors now require for cyber access: single sign-on, MFA, role-based access, logging, hardware keys, incident response documentation
- → A plan for the day a model is withdrawn, re-gated, or replaced

Each line is a software component with an owner, a backlog, and a maintenance cost. Together they are a product, and the model is the part that changes most often and matters least to the evidence. Independent research on thirteen open-source autonomous pentesting frameworks, published in April 2026, found the failure modes a homegrown effort meets first: findings claimed without an executed proof, results that cannot be reproduced, and no record of what was not tested.

What we learned building ours

ELTON built its pipeline by encoding thirteen years of manual medical device testing into a harness, and the honest account is that the model was the easy part. The hard parts were the coverage record, the hardware tooling, the verification layer that turns candidates into evidence, and the operational controls that a vetting program now audits. The pipeline runs five agentic loops writing tools around the clock, and the toolchain holds thousands of device-specific tools, all of it included in a subscription because capability turns into more testing on the customer's behalf rather than a change order.

A manufacturer can build this, and the decision is whether it wants to own a security software product alongside a medical device, with a team to match, or buy the evidence.

The purchase is two things: discovery and management

The most common buying mistake of 2026 is treating AI pentesting as only a faster pentest, because the speed brings a volume of output that a legacy program has no way to absorb.

The volume problem, with numbers

Claude Mythos Preview identified 271 of the 423 security fixes Mozilla shipped in Firefox 150. HackerOne reported a 210 percent rise in AI-generated vulnerability reports in 2025, with autonomous agents submitting more than 560 valid ones. On the curl project, Mythos reported five confirmed vulnerabilities; the maintainer confirmed one, a low-severity CVE. NIST's April 2026 update to the National Vulnerability Database addressed record CVE growth that the disclosure pipeline could not keep up with.

Two things are true at once. Frontier models find real bugs at a rate no team can match. And the majority of what they surface, on any given device, is unreachable, duplicated, or wrong in a way that only an executed test can show. Raw findings volume without context is a liability in a regulated product, unfit for either a reviewer or an engineering backlog.

The analysts who followed the OpenAI and Anthropic releases put it in one sentence: AI compresses the time to discover candidate issues, and the scarce work is now validation, prioritization, safe patching, and retesting, which is exactly the job of the management platform.

Vulnerability versus weakness

A vulnerability is exploitable today, reachable from the real attack surface, and substantiated by a test case generated and executed against the target. A weakness is latent: real, worth tracking, and not exploitable today, so not worth interrupting a release for. Exploitability is answered at runtime on the device, and source, an SBOM, and a diagram are at best a hint toward that answer.

Keeping the two apart is what gives engineering a backlog it can act on.

Verification in three levels

Every candidate finding moves through graduated verification, and each level clears a share of what enters it:

- → **L1, digital twin.** Reachability analysis against the modeled architecture. About 30 to 40 percent of candidates come back Not Affected.
- → **L2, code and firmware.** Static and dynamic analysis on the implementation. About 50 to 65 percent Not Affected.
- → **L3, real hardware.** Exploitation attempted on the running device. About 70 to 85 percent Not Affected.

Every Not Affected disposition carries the evidence for why, which is the first thing a reviewer asks for. The findings that survive L3 are the ones engineering fixes this sprint.

From hundreds of findings to a few root causes

Findings that survive verification are chained through a dependency graph, because a hundred exploitable conditions on a device usually resolve to a few root causes. Fixing the root cause closes the chain. The graph is also what separates a weakness from a vulnerability: it shows whether any entry vector reaches the vulnerable function on this device, as deployed.

The fix, and the proof the fix worked

For each confirmed finding, remediation is written to the file and line, scoped to the actual blast radius, and delivered as a ticket into Jira or Azure DevOps. After engineering ships the fix, the same test case runs again and the finding closes with evidence, without a new engagement. VEX statements, a living SBOM, and the finding lifecycle metrics regulators ask for are generated as a byproduct of every step. FDA recommends tracking "duration from vulnerability identification to when it is updated or patched," and that number only exists when the loop is closed.

The evaluation criteria that matter in 2026

Criteria should be drawn from real compromise paths and from what a reviewer examines. Buyers should look past feature lists and ask whether an approach demonstrates exploitability on the device, records coverage, and closes the loop between discovery and verified fix.

Exploitability proven on the device

A count of findings says little about exposure compared with whether each finding is reachable from a real entry vector and was demonstrated on the running device. Ask whether the vendor executes the test case, on what access level, and what evidence is stored.

Coverage as a written record

Ask for the test plan before the engagement and the coverage record after. Both should be the same threat register. If coverage is described as a percentage of the device with no register behind it, it is an opinion.

Traceability a reviewer can walk

Every finding and every dismissal should trace from the twin through the threat, the test case, and the result to the VEX statement. Section 524B reviewers ask for testing traced to the threat model and a rationale for every threat not tested.

Fix validation without a new statement of work

Ask how a specific fix is retested, how quickly, and whether the retest is targeted rather than a full re-engagement. Exposure persists for as long as verification waits.

Response to a new CVE or a Known Exploited Vulnerability

Emerging threats do not wait for the next engagement. Ask how quickly a newly published CVE or a KEV entry is mapped to the device, verified, and dispositioned. A mature program answers in hours: automated verification within four hours, an early-warning determination within 24, a defensible report within 72, which is the cadence EU CRA and NIS2 timelines require.

Model transparency and the data boundary

Ask which model class runs the pipeline, who holds the access, what controls sit on it, and exactly what enters a model. A vendor who cannot answer the last question in one sentence has not drawn the boundary.

Regulatory scope, stated exactly

The approved claim scope is narrow and specific: ELTON meets FDA premarket (Section 524B) and postmarket cybersecurity vulnerability testing and management requirements. Be wary of any vendor who broadens that into "FDA compliant" or "FDA approved." FDA does not approve testing vendors or tools. It qualified the MITRE CVSS rubric for medical devices under the MDDT program, and a vendor should say "qualified," attribute it to the rubric, and stop there.

An effective evaluation confirms that an approach delivers:

- → **Proven exploitability.** Demonstrated on the device, with stored evidence
- → **Coverage as a record.** A threat register with a result per row, every release
- → **Traceability.** Twin to threat to test to result to VEX, walkable in both directions
- → **Closed-loop validation.** Targeted retest of a fix without a new engagement
- → **New-CVE responsiveness.** Hours to disposition, with the CRA and NIS2 clock in mind
- → **A drawn data boundary.** One sentence on what enters a model, and a model that can be swapped

If a vendor cannot demonstrate these, the program will produce activity without producing evidence.

Questions to ask vendors, which most buyers don't

Procurement usually compares features, prices, and sample reports. The difference between an impressive demo and a defensible program comes down to the questions the buyer is willing to ask.

> *"Show me the test plan before the first packet."*

A coverage record that exists after the test is a findings list with a coverage percentage stapled on. Ask for the threat register the test plan was derived from, and ask what was deliberately not tested and why.

- → Where does the plan come from, the device or the tester?
- → Can I read it before the engagement starts?
- → What does the record look like for a threat that returned nothing?

> *"Which model wrote this tool, and what did it see?"*

A vendor running frontier models should be able to say which class of model built the tooling, under which access program, and with what controls.

- → Which access programs is the vendor approved under, and for what use?
- → What enters a model, in one sentence?
- → Can I choose an open-weight model, and what changes in the deliverable if I do?

> *"What happened to your program on June 13?"*

Fable 5 and Mythos 5 went dark for every customer for eighteen days this summer, and only the vendors with a model-agnostic pipeline kept testing through it.

- → Is the model a configuration or an architecture?
- → What is the plan for the next withdrawal, re-gating, or replacement?

> *"What happens to the 99 percent?"*

Frontier models surface far more candidates than a device has real vulnerabilities. Ask what the vendor does with everything that does not survive verification.

- → Is every dismissal evidenced, or does the report just stop?
- → What share of candidates comes back Not Affected at each verification level?
- → Does a weakness get tracked, or dropped?

> *"How do you retest a fix without a new SOW?"*

Verification is where risk is reduced. Ask how a specific finding is retested after engineering ships a change.

- → Is the retest targeted to the finding, or is it a new engagement?
- → How is the retest documented in the finding lifecycle?
- → How long does a fix stay unverified in practice?

> *"Show me a redacted fix from a comparable device."*

Ask for a redacted remediation from a comparable device.

- → Is it written to the file and line?
- → Is it scoped to the blast radius on this device, or generic?
- → Was it verified before delivery?

> *"How fast can you disposition a KEV entry against my device?"*

Known Exploited Vulnerabilities are weaponized risk. A program that treats them like any other CVE will be late.

- How quickly is a newly listed vulnerability mapped to the device model?
- Is exposure validated before patching and remediation confirmed after?
- Can the early-warning determination land inside the CRA and NIS2 windows?

Clear answers to these questions show how much of a vendor's testing actually reduces exposure.

What effective testing looks like operationally

An effective program runs inside the product's release rhythm and operates as a standing control.

Testing rides the release cadence

Whenever firmware changes, a library is bumped, or a new interface ships, a mature program responds this way:

- Testing runs on every release
- A release is diffed against the last one, and what changed is retested while unchanged evidence carries forward
- Findings land in engineering's own tools as tickets, tied to the component that owns them

Security does not wait for the next scheduled engagement to learn what a release changed.

Findings move through a closed loop

Discovery alone reduces nothing. An effective program enforces one loop for every finding:

- A candidate is identified and verified on the device
- A fix is written to the file and line and shipped by engineering
- The same test case runs again and the finding closes with evidence

The last step is what turns remediation from an assumption into evidence, and it ends most of the severity arguments between security and engineering.

Metrics that reflect exposure

Mature programs measure whether exposure is falling, and the two numbers FDA points at are the ones to track:

- **Mean time to triage (MTTT).** Discovery through ingestion to an initial device-adjusted rating
- **Mean time to remediate (MTTR).** Verification through confirmed fix, or a stop at Not Affected
- **Coverage per release.** Threats tested over threats applicable, as a record
- **Not Affected rate by level.** How much of the volume the verification layer is absorbing
- **Reoccurrence.** Whether a closed finding comes back in a later release

These numbers shift the leadership conversation from how many findings a report had to whether the device is safer than it was last quarter.

Evidence lands in the submission

Every test, verification, and confirmed fix generates the artifacts a submission and a postmarket file need: test cases traced to threats, dispositions with evidence, MDDT-rubric ratings, VEX statements, and a living SBOM. Compliance becomes a byproduct of the program rather than a separate workstream assembled in the weeks before filing.

Common buying mistakes in 2026

Experienced security leaders fall into predictable traps, most of them inherited from what a pentest used to be for.

Buying the model name

A vendor names a frontier model and the evaluation stops there. Two pipelines using the same model produce different evidence, and the model is the component that will change first. Evaluate the harness, the coverage record, and the credential controls, and treat the model as a line item.

Confusing a prompt with a pipeline

A live demo in which a model finds a bug on a firmware image proves only that models find bugs, and leaves coverage, reproducibility, and evidence unaddressed. Ask for the test plan and the dismissal record.

Accepting a findings count as coverage

A long report reads as thorough. Coverage is a register of what was tested and what was not, with a result per row. If the vendor cannot produce that register, the count is measuring effort.

Buying discovery without management

AI-scale discovery without a verification and remediation platform moves the bottleneck onto your engineers. The findings arrive faster than anyone can triage, and the program stalls at the same place it did before, with more paper. Budget for both pillars or neither.

Building it because prompting looked easy

An afternoon with a model produces a prototype. A pipeline a reviewer can trust is a software product with a backlog, an on-call rotation, and a vetting program to satisfy. Decide whether the company wants to own that product, on purpose, before the prototype becomes an obligation.

Treating the pentest as a submission checkbox

A premarket engagement satisfies a filing date, while Section 524B and the postmarket guidance describe a continuous obligation, so a program designed around the filing produces one artifact and a year of exposure.

Letting device artifacts into a model you do not control

Firmware, source, and architecture documents are the most sensitive things a manufacturer owns. Any vendor, or internal team, should state in one sentence what enters a model and what never does, and be able to prove it. If the answer involves a policy rather than a data path, keep asking.

How to apply this inside your organization

Agreeing with a framework is easier than executing it, and four steps make the difference.

Assess the current program honestly

Begin by examining:

- How often the device is tested, and whether the cadence follows the release calendar or the filing calendar
- What share of the device's testable areas the last engagement covered, as a list
- How exploitability was demonstrated, and on what access level
- How fixes were validated, and how long they stayed unverified
- How quickly a newly published CVE reaches a disposition
- What entered a model, if anything, and on whose infrastructure

Then ask the harder question: has the current approach measurably reduced the device's reachable exposure since the last release? If the answer is unclear, the evaluation criteria need to change before the vendor does.

Run a realistic evaluation

A demonstration is built to highlight strengths, so the evaluation has to be built to reveal limits. When assessing a vendor:

- Put a real device on the bench, with real firmware, rather than a sample application
- Ask for the test plan before the run and the coverage record after
- Read a Not Affected disposition and check that the evidence would survive a reviewer
- Time the path from a new CVE to a disposition on your device
- Ask for a fix on a real finding and see whether it names a file and a line

The goal is to observe the program under the conditions it will actually run in, including the day a model is withdrawn.

Involve regulatory and engineering early

Testing outcomes land on two teams other than security. Regulatory affairs has to defend the evidence, and engineering has to ship the fixes and live with the backlog. Early involvement matches the evidence format to what a submission needs and puts remediation into the tools engineering already uses.

Justify the budget on exposure and evidence

Executive conversations default to breach scenarios. A more durable case is made on exposure and evidence:

- → Fewer findings, and less severe ones, appearing in formal regulatory review
- → Shorter time from discovery to verified fix, tracked as a number
- → A coverage record that grows with every release
- → A defensible answer, in hours, when the next KEV entry hits a component in the SBOM

Evidence-based arguments outlast fear-based ones in the room, and they also survive in front of a reviewer, which is where the budget is ultimately tested.

Choosing what produces evidence

The point of a testing strategy is to make a device harder to compromise and to prove it, release after release, to a regulator who will ask.

In 2026 the buyer needs more than findings and a formatted report: a coverage record derived from the device, exploitability demonstrated on it, dismissals with evidence, fixes with a file and a line, and a retest that closes the loop. Behind that sit the strongest discovery models available, run under credentials and controls the manufacturer does not have to build and kept swappable, plus the management platform that reduces machine-scale discovery to a short backlog with a proof attached to everything it dismissed.

Trade-offs remain, since a consulting engagement can answer a filing date and a scanner suite will catch the obvious in CI. But a manufacturer whose goal is measurable, evidenced reduction of reachable exposure across the supported life of a device will look for a continuous program, a platform delivered as a managed program, one that runs the tests, verifies what they find, guides the fix, confirms it landed, and can name in one sentence what its models see.

The reviewer keeps proof, which a program produces release after release and a pentest never did.

Glossary

Exploitability. The degree to which a weakness can be used to gain access, escalate privilege, move laterally, or reach protected data on the device as deployed. Answered at runtime on the device, by demonstration.

Vulnerability. A finding that is exploitable today, reachable from the real attack surface, and substantiated by a test case generated and executed against the target.

Weakness. A latent finding that is real and worth tracking but not exploitable today. Tracked, not urgent.

Digital twin. A model of a release's security architecture, built from the manufacturer's quality-system artifacts, used to derive the threat register, judge reachability, and carry evidence forward across releases.

Threat register. The list of threats that apply to the components of the twin, on the MITRE EMB3D backbone. With a test case on each row it is the test plan, and with results filled in it becomes the coverage record.

Verification levels. L1 on the digital twin (reachability), L2 on code and firmware (static and dynamic analysis), L3 on real hardware (exploitation attempted on the running device). Each level clears a growing share of candidates as Not Affected, with evidence.

Dependency graph. The chained model of exploitable conditions on a device, used to distill hundreds of findings to a few root causes and to show whether any entry vector reaches a vulnerable function.

Known Exploited Vulnerability (KEV). A vulnerability confirmed to be actively exploited in the wild. Weaponized risk, requiring faster disposition and remediation than a general CVE.

VEX. Vulnerability Exploitability eXchange. A machine-readable statement of whether a product is affected by a vulnerability, with justification. Generated as a byproduct of every disposition in a mature program.

MDDT. FDA's Medical Device Development Tools program. FDA qualified the MITRE CVSS rubric for medical devices under MDDT (Q171974). Ratings on the rubric are device-contextual and defensible in review.

Cyber Verification Program. Anthropic's verified-defender pathway giving vetted US organizations reduced cyber safeguards, including access to Claude Mythos 5.1, for defensive security work.

Daybreak Red. The tier of OpenAI's Trusted Access for Cyber that provides GPT-5.6 Cyber to approved organizations for authorized penetration testing, red teaming, exploit validation, and controlled vulnerability research.

Credentialed operator. A vetted testing provider that holds these approvals and runs the models inside its own controlled pipeline on the manufacturer's behalf, so the manufacturer receives the evidence without obtaining the model.

Evaluation checklist

Use these questions in vendor evaluations or in an honest review of an internal program. Unclear or inconsistent answers mean the framework is incomplete.

Exploitability

- → Is every finding demonstrated on the device, with stored evidence?
- → Does the vendor distinguish a vulnerability from a weakness, with reasoning?
- → What share of candidates comes back Not Affected at L1, L2, and L3?

Coverage

- → Is there a written test plan, derived from the device, before the first packet?
- → Is the coverage record the same register with a result per row?
- → Does coverage carry forward across releases, or reset?

Traceability

- → Can a finding be walked from twin to threat to test case to result to VEX?
- → Is there a rationale for every threat not tested?
- → Are ratings on the MDDT-qualified rubric, with the evidence attached?

Fix validation

- → Is remediation written to the file and line, scoped to this device?
- → Is a fix retested with the same test case, without a new engagement?
- → How long does a fix stay unverified in practice?

Response to new CVEs and KEVs

- → How quickly is a newly published CVE mapped to the device and dispositioned?
- → Can an early-warning determination land inside the CRA and NIS2 windows?
- → Is exposure validated before patching and remediation confirmed after?

Models and data

- → Which access programs is the vendor approved under, and for which use cases?
- → What enters a model, in one sentence?
- → Can the manufacturer choose frontier, open-weight, or its own model per engagement?
- → What is the plan for the day a model is withdrawn or re-gated?

Program

- → Is the delivery a platform delivered as a managed program, with hardware on the bench?
- → Are the six subscription requirements met, in writing?
- → Is the FDA claim stated in the approved scope, and nothing broader?

Sample program metrics

Mature programs track measurable indicators rather than findings counts. Consider monitoring:

- → Mean time to triage (MTTT)
- → Mean time to remediate (MTTR)
- → Coverage per release, as threats tested over threats applicable
- → Not Affected rate at each verification level
- → Reoccurrence rate for closed findings
- → Time to disposition for newly listed Known Exploited Vulnerabilities

Together they show whether reachable exposure is falling over time and whether the improvement holds.

FDA, Cybersecurity in Medical Devices: Quality System Considerations and Content of Premarket Submissions (SPDF and TPLC language quoted verbatim). Anthropic, "Introducing Claude Fable 5.1 and Claude Mythos 5.1," September 2026. OpenAI Help Center, "OpenAI Daybreak, Trusted Access for Cyber Overview." VentureBeat, "OpenAI launches GPT-5.6-Cyber," August 10, 2026. Software Analyst Cyber Research, "The Cybersecurity Implications of Claude Mythos and OpenAI Cyber," May 11, 2026. Penligent, "OpenAI Daybreak vs Anthropic Mythos," May 14, 2026, and "GPT-5.4 Cyber vs Claude Mythos," April 17, 2026. "Hackers or Hallucinators?", SoK on LLM-based automated penetration testing, April 2026. ELTON basis of claim: 5 of 24 areas, about six

weeks, about $100,000, and fewer than ten true positives reflect the median of consulting engagements ELTON has reviewed and quotes verified across three firms; 7 of 40 and 34 of 40 reflect ELTON program data; verification Not Affected rates are ELTON program ranges.

About ELTON. ELTON is the exploitability management platform for medical devices, delivered as a managed program. ELTON builds a digital twin of each device, runs the pipeline across firmware, protocols, and real hardware every release, verifies which findings are exploitable on the device, and evidences every finding and every dismissal for regulatory review. ELTON meets FDA premarket (Section 524B) and postmarket cybersecurity vulnerability testing and management requirements. Talk to us at **eltoncyber.com**.