ELTON

WHITEPAPER

Continuous Cybersecurity Testing and Vulnerability Contextualization

The future of medical device testing: why a product that changes every sprint cannot be secured by an engagement that happens once a year

KEY TAKEAWAYS

- → Volume is not a result. A test produces hundreds of raw findings, dozens are real only behind another root cause, and a few are directly exploitable. Those few carry the regulatory exposure, and telling them apart takes context a two-week engagement never has time to earn.
- → FDA asks for frequent testing in plain words. The premarket guidance puts cybersecurity testing "throughout the SPDF" and, after release, "at regular intervals commensurate with the risk." It also asks manufacturers to report how long a vulnerability stays open. A once-a-year engagement is what makes that number bad.
- → Seven drivers make continuous testing the better model, not only the compliant one: lower regulatory risk, lower cost, faster milestones, reserved capacity, more depth, faster V&V, and an AI discovery pipeline that pays off only inside a platform that classifies what it finds.
- → Six questions separate a continuous model from an annual engagement with a subscription invoice. Each one is answerable with evidence, and each one is listed at the end of this paper.

Executive summary

A connected medical device changes every sprint. Its software bill of materials changes with it, and new vulnerabilities are published against its components every week. The traditional security engagement looks at that product for two or three weeks, once a year, and produces a report that starts aging the day it is signed.

The cost of that mismatch is not only the price of the engagement. It is the findings that surface at the gate instead of in development, the release that waits on a consultant's calendar, the regression testing forced by remediation advice nobody can act on, and the third-party vulnerability report that lands because a researcher spent six months on a device the tester saw for two weeks.

Continuous testing fixes the mismatch by changing what is kept between tests. A model of the product, the digital twin, records what changed between releases. Every test case ever run is retained and re-runnable. Each finding is placed in the product's exploit chain, the dependency graph, and classified by whether an attacker can actually reach it. The next test only has to cover what moved, and the evidence from everything that did not move carries forward.

This paper sets out the problem, the FDA expectation behind it, the seven drivers for a continuous model, the remediation gap it closes, and what to require of any testing partner that claims to offer it.

1. The problem: hundreds of findings, a few that matter

A security test of a connected device does not produce a short list. Scanners, SBOM feeds, code analysis, fuzzing and manual testing together produce hundreds of raw results. Most of them describe a real weakness in a real component. Very few describe something an attacker can use on this device, as shipped.

The work that matters is classification, and it sorts every finding into one of four verdicts:

VERDICT	WHAT IT MEANS	HOW MANY THERE SHOULD BE
Directly exploitable	Exploitable on its own from a known entry vector, with every precondition met by the product as shipped	A handful at most: the number under regulatory exposure.
Conditionally exploitable	Real, but reachable only while another root cause keeps producing the condition it needs	Dozens. Fix the root and the condition closes.
Unexploitable weakness	No path on this device today	The majority. Ships dismissed, with its reasoning attached, and is re-evaluated as the product changes.
Mitigated weakness	A path that existed has been closed by a fix or control, and the retest proves it	Grows over time, as the record of work done.



Figure 1. From results to findings. Only a verified exploit path turns a result into a finding. Everything else is recorded as a weakness with the reason attached.

Classification takes context: which interfaces reach the vulnerable component, which trust boundaries an exploit crosses, which controls sit in the way, what changed since the last release. That context is earned over time. A point-in-time test calls everything a vulnerability because two weeks with a device is not enough time to know what is reachable.

That is also why third-party vulnerability reports land so hard. The researcher who found the flaw had context the tester never bought.

2. Frequent testing is what FDA asks for

This is not a preference argument. FDA's premarket cybersecurity guidance builds the submission on a framework that runs the length of the product, and it says in plain words when testing should happen.

> *"An SPDF encompasses all aspects of a product's lifecycle, including design, development, release, support, and decommission." Section V, Secure Product Development Framework*

> *"FDA recommends that cybersecurity testing should occur throughout the SPDF. Security testing early in development can ensure that security issues are addressed prior to impacting release timelines and can*

> *prevent the need to redesign or re-engineer the device. After release, cybersecurity testing should be performed at regular intervals commensurate with the risk (e.g., annually)." Section V.C, Cybersecurity Testing*

The guidance ties the same expectation to the total product life cycle:

> *"Security risk management should be an integrated part of a manufacturer's entire quality management system, addressed throughout the TPLC." Section V.A*

> *"Cybersecurity risks may continue to be identified throughout the device's TPLC. Manufacturers should ensure they have appropriate resources to identify, assess, and mitigate cybersecurity vulnerabilities as they are identified throughout the supported device lifecycle." Section V.A.6, TPLC Security Risk Management*

FDA also asks manufacturers to report the clock. Section V.A.6 recommends tracking and submitting the "duration from vulnerability identification to when it is updated or patched." A once-a-year engagement is what makes that number bad: a vulnerability introduced in March and found in the January test has already been open for ten months before anyone starts fixing it.

The annual interval in the guidance is the floor, given as an example. The seven drivers below are the case for going past it.

3. Seven reasons continuous beats point-in-time

	DRIVER	IN ONE LINE
01	Lower regulatory risk	Findings become development artifacts you control, not report entries you have to explain.
02	Lower cost	Unchanged components carry forward on evidence. You pay to test the diff, not the product.
03	Faster milestones	Milestone tests close in under two days, and reports never go stale.
04	Dedicated capacity	Capacity is reserved in advance. No lead time, no queue, no expedite fee.
05	More depth	Coverage compounds across months instead of restarting at zero every engagement.
06	Faster V&V	Contextual fixes go straight to developers, scoped to the smallest safe change.
07	AI needs a platform	AI finds more, faster. Without contextualization against the real attack surface that is only a larger pile.

Driver 01: reduce regulatory risk

Continuous testing changes what a vulnerability is, in the regulatory sense. Found during development, it is an engineering task. Found during the formal exercise, it is a record.

A vulnerability discovered during development never enters a formal report or a quality document. It is a development artifact, and the manufacturer controls the timeline on it: no explanation owed to a reviewer, no customer-facing disclosure, no clock someone else set. Continuous discovery drains that backlog before the formal test begins, so the report that reaches FDA carries fewer findings at lower severity, and the customer-facing reports raise fewer questions on every deal and every audit. That is also what Section V.C describes: finding and fixing inside development, not at the gate.

The goal is not a better report. It is a formal test that has very little left to find. Every finding that reaches a submission or a customer report costs something: review time, a justification, a remediation commitment, sometimes a clearance delay. The cheapest finding is the one your own engineers closed six months earlier.

Driver 02: reduce the cost of testing

When testing keeps pace with the product, the cost of the next test is marginal. That only works if someone kept the evidence from the last one.

In the subscription model, the traceable test cases from every prior test are retained and re-runnable. The digital twin tracks what actually changed between releases, so each test diffs the twin, zeroes in on the change and tests that. The hundreds of test cases already run against components that did not change become the evidence that no further testing was required.

Legacy consulting starts from scratch. A new consultant enters the project each time, the process restarts for a product that changed incrementally, and the manufacturer pays for the overhead of restarting: scoping, onboarding, a person re-learning the product. Nothing from last year's report reduces this year's work.

Evidencing that a component needs no new testing requires a platform that holds the prior test cases, the software model and the diff between them. That is a subscription capability. A statement of work cannot produce it.

Driver 03: shorter test windows, faster releases

Two separate effects are at work. One shortens the test. The other extends how long its result stays valid.

Testing that keeps pace only has to cover the increment, not the whole product, so a milestone test closes in under two days instead of a four to six week engagement. Security testing moves inside the sprint cadence instead of sitting between the sprint and the launch, and nothing waits on a consultant's availability because the capacity is already reserved (Driver 04).

Reports stop going stale. Incremental testing plus continuous SBOM monitoring means today's vulnerability set is known today. If the software has not changed since the last test and new public CVEs are accounted for, the report can be regenerated on demand. That outlasts the roughly six month shelf life a one-time manual test carries into a submission.



Figure 2. A pentest is evidence for the day it ran. The subscription keeps the evidence current: every new CVE is correlated against the test results, and the report regenerates on demand.

A point-in-time report starts decaying the day it is signed. Every CVE published after the test date is unaccounted for, and the only way to answer for it is another engagement. A continuously maintained model answers the same question by regenerating the report against the current CVE set.

Driver 04: dedicated capacity, no lead time

An annual subscription reserves the team before it is needed. Release schedules move constantly, and the testing is still there on the day the schedule lands.

Consulting capacity is sold by the week to whoever booked first. A code freeze that slips means requeuing, and the next open slot is weeks out. Pulling testing forward costs a short-notice or prioritization fee, charged exactly when the manufacturer has the least room to argue, and testing gets scheduled around the consultant instead of around the release.

Reserved capacity removes all of that: no lead time and no availability check when a milestone moves, no expedite fee, no surge rate in the week that matters. The answer to "can you test this Thursday" is yes, because the capacity is already yours. "We will have to check the schedule" is one of the most expensive sentences in a release plan, and nothing about it is a security finding.

Driver 05: greater depth over time

Two to three weeks is a ceiling on how deep a paid engagement can go. Independent researchers and large hospital systems do not have that ceiling.

A point-in-time engagement spends its budget re-covering the same surface every year, so depth never accumulates. Researchers and high-profile healthcare delivery organizations spend far longer on a single device, with no monetary constraint, and that asymmetry is exactly why third-party vulnerability reports are so damaging.

Continuous coverage compounds instead. Each cycle targets what was not covered, or was not the focus, last time. Test case execution history builds across the product over months and years, breadth and depth both increase at a lower total cost than any single engagement, and after a year the tester knows the product better than the last consultant ever could.

The manufacturer is not competing with the consultant's two weeks. It is competing with the researcher's six months, and the only affordable way to close that gap is to keep testing the same product for years.

Driver 06: cut V&V time with optimized fixes

Generic remediation guidance forces broad regression testing. A fix written against the actual product does not. Section 4 shows the difference in detail.

Two capabilities make the saving. Posture tracking finds the single fix that closes the most open findings, clearing the high and critical set at once instead of one ticket at a time. And because the twin shows what the fix touches, the V&V scope is argued from evidence rather than assumed, so it shrinks instead of expanding. The guidance is written to the product, so it goes straight to the development team without translation. ELTON's remediation optimization is built on exactly this.

Driver 07: AI discovery only pays off inside a subscription

AI compresses discovery to days. It also produces far more output than a human triage pass can absorb, which is why the platform matters more than the model.

In ELTON's agentic pentesting, frontier models write and run the tools, so a test finishes in days instead of weeks. The output is large, and raw volume is not a result: more findings with no context is a worse deliverable, not a better one. A vulnerability management platform is what determines exploitability, assigns the CVSS rating and classifies each finding against the real attack surface. Code that cannot be reached from the attack surface outside the case is a backlog item, not an emergency, and the four verdicts are what make the final number defensible. On the physical device, TestLink is what carries that pipeline out of the lab simulation and onto the real hardware.

The value also has to be shared. A subscription passes the efficiency back to the manufacturer as more testing and better results. A point-in-time engagement keeps the long turn times and tends to stay quiet about where AI is used. Ask any tester where AI sits in their process and what it changed about your price.

A medical device under regulatory exposure should not carry more than a handful of directly exploitable vulnerabilities. If a report says otherwise, the more likely explanation is that the tester never had enough context to say what was reachable.

4. The remediation gap

The difference between consulting guidance and a fix written to the product is easiest to see side by side.

CONSULTING GUIDANCE	THE ELTON FIX
F-01 SQL injection in device query: "You should sanitize input."	Weakness: TLS negotiation on the management port. Remove ctx = ssl.PROTOCOL_TLS; add ctx.minimum_version = ssl.TLSVersion.TLSv1_3.
F-02 Weak TLS on management port: "Perform better encryption."	Verified against the digital twin at runtime, then delivered into Jira or Azure DevOps.
No file, no line, no patch, no verification. Development guesses, then V&V retests everything the guess touched.	V&V scopes to the line that moved. The retest is the same test case that found the issue, so the closure is evidence, and the finding becomes a mitigated weakness.

The second column is not a better-written report. It is what becomes possible when the tester holds a model of the product, the code and the test case together, and keeps them between releases.

5. What to require of a testing partner

Six questions separate a continuous model from an annual engagement with a subscription invoice. Each one has an answer a buyer can check.

	REQUIRE	WHY IT MATTERS
01	Test case traceability that carries across engagements, not a fresh scope each time	Without it, nothing from the last test reduces the next one.
02	A model of the product that shows what changed since the last test, at component level	The diff is what lets unchanged components carry forward on evidence.
03	Findings classified as directly exploitable, conditionally exploitable, unexploitable weakness or mitigated weakness, with the reasoning attached	The classification is what a reviewer will ask for, and what keeps the directly exploitable set to a handful.
04	Fixes written to a file and a line, verified before they reach a developer	This is the difference between a two-day V&V cycle and a full regression.
05	Reports that can be regenerated against today's CVE set without a new engagement	A report that cannot be regenerated is stale six months after it is signed.
06	Transparency on where AI is used, and how that efficiency reaches your price and your coverage	AI should buy the manufacturer more testing, not the tester a better margin.

Continuous testing is not a faster version of the old engagement. It is the only model where evidence accumulates instead of being rebuilt every year. The continuous testing subscription page shows how ELTON delivers it.

FDA, Cybersecurity in Medical Devices: Quality Management System Considerations and Content of Premarket Submissions, final guidance, February 2026, Docket FDA-2021-D-1158, Sections V, V.A, V.A.6 and V.C. Section 524B of the Federal Food, Drug, and Cosmetic Act. Timing, cost and capacity figures in this paper (under two days per milestone test, four to six week engagements, two to three weeks of testing per engagement, a roughly six month report shelf life) describe ELTON's delivery experience and typical consulting engagements; they are not FDA figures. The fix examples in Section 4 are illustrative.

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**.