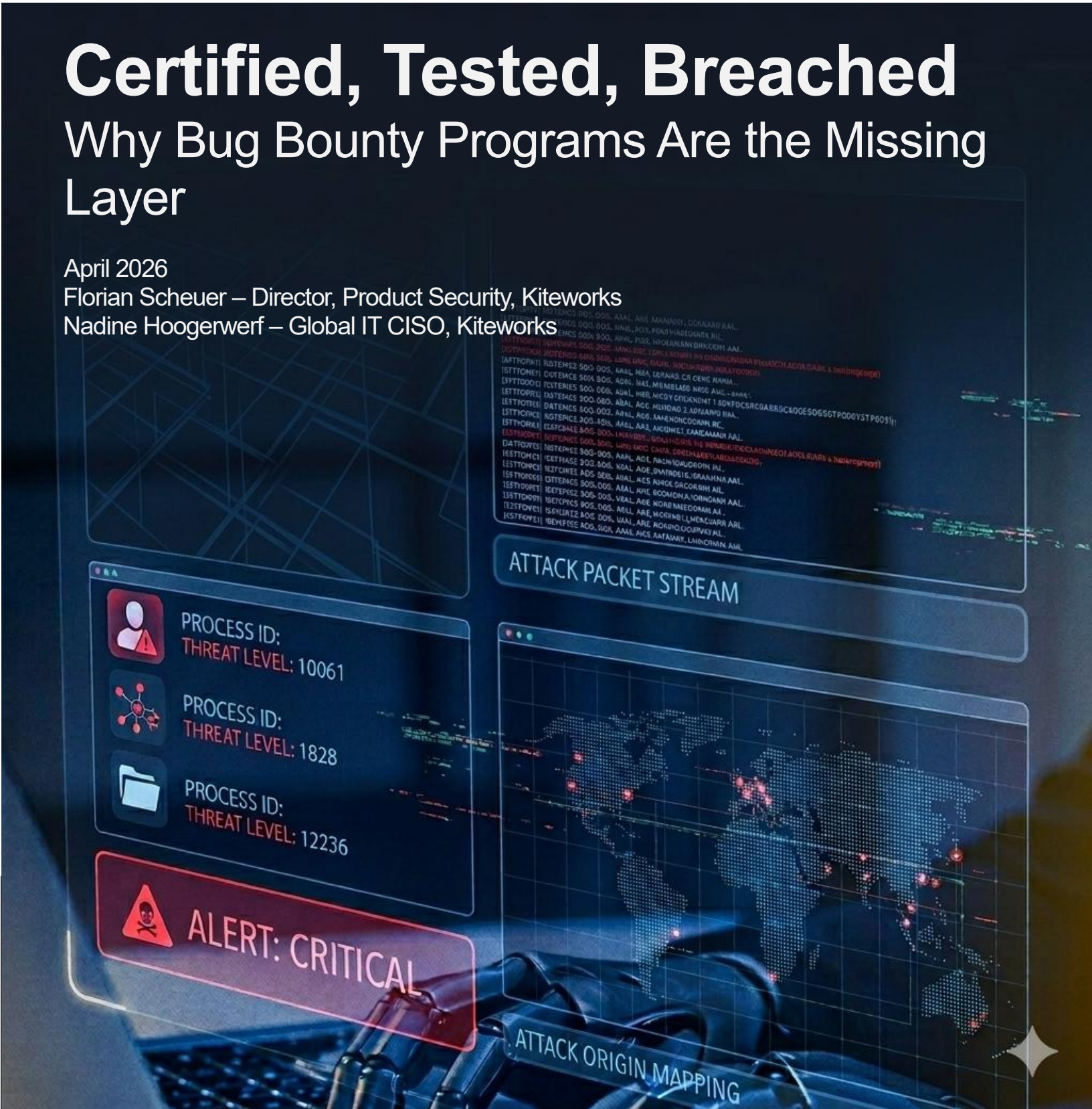
Kiteworks

Certified, Tested, Breached

Why Bug Bounty Programs Are the Missing Layer

April 2026
Florian Scheuer – Director, Product Security, Kiteworks
Nadine Hoogerwerf – Global IT CISO, Kiteworks



Executive Summary

Software security is not a certificate. It is not a report. It is the outcome of continuously exposing a product to the same scrutiny an attacker would apply — and acting on what that scrutiny reveals.

Most vendor security conversations stop well short of that standard. Compliance certifications, while genuinely valuable, are designed to assess whether an organization operates with security-conscious governance — not whether its products can withstand an active attempt to compromise them. Penetration tests go further, but remain bounded by fixed timelines, negotiated scope, and a single team's perspective. A test that concludes on a Friday tells you nothing about the vulnerability that ships on Monday.

Attackers operate differently. They are patient, specialized, and creative. They are not constrained by scope agreements or reporting deadlines. They will spend months building familiarity with a target, pursuing unconventional paths, and chaining together observations that no time-boxed engagement would have had the opportunity to connect. They only need to find one way in — and they have as long as they need to find it.

Bug bounty programs are the testing model that reflects this reality. By opening a product to a global community of self-selected, specialized researchers — and paying meaningfully for valid findings — vendors create a continuous, adversarially realistic scrutiny that point-in-time testing cannot replicate. Researchers bring deep expertise, invest significant time, and are incentivized to find genuine impact rather than fill a report. The result, over time, is a product that has been continuously hardened against the conditions it will actually face.

For security leaders evaluating vendors, this distinction is actionable. Compliance certificates and penetration test reports should remain part of due diligence — but they are the baseline, not the differentiator. The question that separates vendors genuinely committed to security from those committed to its appearance is whether they have invested in continuous adversarial scrutiny: whether they run a bug bounty program, how long it has been running, how broadly it is scoped, and whether the rewards are meaningful enough to attract researchers capable of finding what others miss.

Kiteworks has built exactly this structure — running multiple bug bounty programs across independent platforms for our core product, and establishing programs for every subsidiary upon acquisition. We apply to ourselves the standard we are asking you to apply to your vendors. Because when the question is eventually asked — not whether a certificate was in place, but whether everything possible was done to prevent a breach — the only answer that holds is one built on all three layers: governance, validation, and continuous adversarial scrutiny.

Contents

1. Introduction: The Question Behind the Question

When a security or procurement team evaluates a software vendor, penetration testing has become the expected answer to the security question. Vendors commission tests, receive reports, and share summaries with prospective customers. Procurement checklists get ticked. Audit requirements get satisfied. The engagement feels rigorous — a team of specialists, a defined methodology, a written report with findings and remediation guidance. The question vendors are asked is whether they have been tested. The question that actually matters is whether their product is secure. These are not the same thing.

And in many respects, it is rigorous. Penetration testing is a legitimate and useful security practice, and organizations that conduct regular tests are demonstrably more security-conscious than those that do not. The problem is not that penetration testing is without value. The problem is what happens when it becomes the primary — or the final — answer to the question of whether a product is secure.

That question deserves a more honest answer than a point-in-time report can provide. Software is not static. Codebases evolve, dependencies change, new features introduce new attack surface, and deployments shift configurations in ways that no previous test could have anticipated. An attacker approaching your vendor's product today is not constrained by the scope or the timeline of an engagement that concluded months ago.

The organizations and vendors that are genuinely ahead of the threat landscape understand this. They treat penetration testing as one input among several — valuable, but bounded. And they invest in mechanisms that go beyond it: continuous, incentive-driven, adversarially-oriented testing that reflects how real attacks actually happen. Chief among those mechanisms is the bug bounty program.

This paper examines why. It looks honestly at what penetration testing delivers and where its structural limits lie. It makes the case for bug bounty programs not as a replacement for existing security practices, but as the layer that closes the gap between a defensible security posture and a genuinely resilient one. And it offers practical guidance for security leaders evaluating vendors — because asking the right questions at the procurement stage is itself a security decision.

2. What Compliance and Certification Actually Deliver

For any organization handling sensitive data or operating critical services, compliance certification is not optional — and rightly so. Frameworks like ISO 27001, SOC 2, or the requirements flowing from NIS 2 and DORA exist because experience has shown that organizations without structured security governance make predictable and avoidable mistakes. Certifications provide evidence that a vendor has done the foundational work: responsibilities are assigned, policies are documented, processes for handling security events are defined, and the organization submits to external scrutiny on a regular basis.

This is not trivial. A vendor without these foundations represents a category of risk that no amount of technical testing can compensate for. Governance failures — unmanaged access rights, undocumented systems, absent incident response plans — create vulnerabilities that are organizational rather than technical in nature. Compliance frameworks are designed to eliminate exactly these failure modes, and they do so effectively.

The ceiling, however, is important to understand clearly.

Certification frameworks assess whether an organization behaves like a security-conscious company. They evaluate processes, documentation, governance structures, and organizational culture. What they do not — and cannot — assess

is whether the software that organization produces or operates contains exploitable vulnerabilities. That is simply not what they are designed to measure.

An organization can achieve and maintain ISO 27001 certification while shipping software that contains a critical authentication bypass. It can pass a SOC 2 audit while running an API that leaks sensitive data under specific conditions. The certification is not wrong in these cases — it is answering a different question than the one that matters most to the attacker.

This is not a flaw in compliance frameworks. It is a boundary condition that security leaders need to hold clearly in mind. Certification is the foundation. It tells you that a vendor takes security seriously as an organization. It does not tell you that their product can withstand the scrutiny of someone actively trying to compromise it.

That question requires a different answer entirely.

3. The Role and Limits of Penetration Testing

Penetration testing occupies a natural position in the security landscape precisely because it appears to answer the question that compliance cannot. Where certification assesses the organization, a penetration test assesses the product. A skilled team attempts to find and exploit vulnerabilities, documents what they discover, and delivers a report with concrete remediation guidance. It is hands-on, technical, and specific.

The value is real. A well-executed penetration test will surface vulnerabilities that internal teams have missed, validate that security controls behave as intended, and produce a written record of findings that can drive a remediation cycle. For many compliance frameworks, a penetration test is also a formal requirement — and the structured output it produces satisfies that requirement appropriately.

The limits, however, are structural. They are not a reflection of the quality of any particular firm or team. They are inherent to the model itself.

Time is the first constraint. A typical penetration test runs for one to four weeks. That window sounds substantial until you consider what a skilled researcher actually needs to do before meaningful discovery can begin. Understanding how a complex product is structured, mapping its attack surface, identifying where trust boundaries exist and how data flows between components — this work takes time, and it has to happen before any serious vulnerability hunting can start. In a two-week engagement, a significant portion of the available time is consumed by onboarding, scope clarification, and the preparation of the final report. The window of actual deep testing may be measured in days rather than weeks. For a mature, feature-rich product, that is rarely enough time to move beyond the more accessible findings.

The talent pool is the second constraint. A penetration test gives you the skills and perspective of the specific team you engaged, applied over a fixed period. Even the best firms have areas of strength and areas of relative weakness. The researcher who would have found the subtle business logic flaw in your file sharing workflow may not be on the team you hired. There is no mechanism in a fixed engagement to compensate for this — the team you contracted is the team that tests your product, and their collective blind spots become your unexamined risk.

Scope is the third constraint. Penetration tests operate within a negotiated boundary. This is understandable and often necessary — unscoped testing creates operational risk and legal ambiguity. But the consequence is that large portions of the actual attack surface may go untested. Attackers do not respect scope agreements. They will probe the integration that was excluded from the engagement, the legacy endpoint that was deemed out of scope, the configuration that nobody thought to include. A finding that falls outside the agreed scope is a finding that does not appear in the report.

The incentive structure is the fourth constraint. Penetration testing firms are engaged for a defined period and paid for their time. There is no financial incentive to find more vulnerabilities or to push deeper into the product — the engagement concludes when the time runs out, regardless of what remains undiscovered. This is not a criticism of professional

integrity; it is simply an observation about how fixed-term engagements work. The model rewards the delivery of a report, not the depth of the investigation.

Taken together, these constraints mean that a penetration test, however competently executed, provides a bounded and time-limited view of a product's security posture. It is a useful input. It is an appropriate response to certain compliance requirements. But it is a snapshot of a moving target, taken through a narrow window, by a team that is still learning the product when the clock starts — and has already begun writing the report before they have truly finished exploring it.

4. How Attackers Actually Work

Understanding why bug bounty programs are effective requires a brief step back to consider the nature of the threat they are designed to reflect. Not in technical detail — but in terms of the fundamental asymmetry that defines every security challenge.

An attacker approaching a software product operates under entirely different conditions than a penetration tester. There is no scope agreement, no time limit, no requirement to document findings in a client-ready format. There is no onboarding meeting and no kickoff call. The attacker simply begins — and continues for as long as the target remains interesting and the potential reward justifies the effort.

Patience is perhaps the most underappreciated factor. Where a penetration test is measured in weeks, a motivated attacker may spend months familiarizing themselves with a product before attempting anything that would register as an intrusion. They read documentation, explore edge cases, observe how the application behaves under unusual inputs, and build a mental model of the system that grows more detailed over time. The vulnerability they eventually exploit may be one that only became visible after that extended period of observation — something no two-week engagement could reasonably have uncovered.

Specialization is the second factor. The attacker community is not a homogeneous group applying a standard methodology. It is a diverse ecosystem of individuals with deeply specialized skills — one researcher who has spent years studying authentication protocols, another who focuses exclusively on file parsing vulnerabilities, a third who understands the specific quirks of a particular framework or runtime environment. Any given product will attract the attention of people whose expertise happens to align precisely with its weakest points. There is no way to predict which specialization will prove most relevant, and no fixed team can replicate the breadth of that collective expertise.

Creativity is the third factor. Real vulnerabilities — particularly the severe ones — are rarely found by following a checklist. They emerge from unexpected combinations: a flaw in one component that becomes critical only when chained with a quirk in another, a design assumption that holds true in all documented cases but breaks under a specific sequence of actions that no tester thought to try. This kind of discovery requires not just skill but genuine curiosity and a willingness to pursue unconventional paths. It is the product of deep familiarity with a target, not a structured methodology applied under time pressure.

The implication for security testing is direct: if the goal is to find what an attacker would find, the testing model needs to reflect how attackers actually operate. Patient, specialized, creative, and unconstrained by artificial boundaries. A model, in other words, that looks much more like a bug bounty program than a penetration test.

5. What Bug Bounty Programs Actually Deliver

A bug bounty program is, at its core, a standing invitation. The vendor opens their product to scrutiny from a global community of security researchers and agrees to pay for valid, impactful findings. The researchers self-select based on interest, expertise, and the belief that their skills are well-matched to the target. The engagement has no end date, no fixed scope negotiation, and no predetermined team.

That structural difference from penetration testing is not incidental — it is the source of most of what makes bug bounty programs effective.

Continuity closes the snapshot problem. A bug bounty program runs while the product evolves. When a new feature ships, it enters scope. When a dependency is updated, it can be examined. When a configuration changes, researchers are already familiar enough with the product to notice when something behaves differently. The program does not need to be restarted or rescoped each time the codebase changes — it simply continues, reflecting the product as it exists today rather than as it existed at the time of the last engagement.

Depth of expertise replaces the fixed team. Rather than the skills of one contracted firm, a bug bounty program draws on a global pool of researchers with diverse and highly specialized backgrounds. The researcher who finds a critical vulnerability in your product may be someone who has spent years focused on exactly the class of issue your architecture happens to be susceptible to. That match between specialized expertise and specific vulnerability class is not something that can be engineered in a fixed engagement — it emerges naturally from the self-selection dynamics of an open program.

Time investment produces qualitatively different findings. Bug bounty researchers are not constrained by a two-week window. A researcher who finds your program interesting may spend weeks or months building familiarity with the product before submitting a single report. That accumulated understanding is what enables the discovery of complex, chained vulnerabilities — the kind that require connecting observations made at different points in time, across different parts of the application. These are precisely the findings that time-boxed engagements are least likely to surface, and precisely the findings that represent the most serious real-world risk.

Incentive alignment produces honest results — but only if the incentives are meaningful. Researchers are paid for valid findings, not for time spent. There is no incentive to pad a report with low-severity observations to justify the engagement fee, and no incentive to stop digging once a reporting deadline approaches. A researcher who suspects there is something significant in a particular area of the application will continue pursuing it for as long as their intuition holds — because the reward for a critical finding is substantially higher than the reward for nothing. The program pays for outcomes, and that shapes behavior in ways that fixed-fee engagements structurally cannot.

But this dynamic only functions if the reward levels are set seriously. Skilled security researchers have choices. They can direct their time and expertise toward any number of programs, and they will naturally gravitate toward those that offer meaningful compensation relative to the effort and complexity involved. A program that offers token rewards for critical findings will attract token effort. A program that pays competitively — that signals to the research community that the vendor genuinely values what they bring — will attract the caliber of researcher capable of finding what others miss. The size and structure of the bounty is not a budget line to be minimized; it is the primary lever that determines the quality of scrutiny the product receives.

The program improves the product over time. This is perhaps the most significant long-term benefit, and the one most relevant to customers evaluating a vendor's security commitment. As findings are reported and remediated, the product becomes progressively more resistant to attack. Systemic weaknesses — architectural patterns that generate vulnerabilities, dependency management practices that introduce risk, authentication designs with recurring flaws — become visible through the accumulation of findings and can be addressed at the root rather than symptom by symptom. A vendor who has run a rigorous bug bounty program for several years is operating a product that has been continuously stress-tested and hardened by that process. That is a qualitatively different security posture than one maintained through periodic testing alone.

For customers, this translates into something concrete and meaningful: a lower probability of being the victim of a breach that the vendor's own testing process should have caught. Not zero — no security program eliminates risk entirely. But meaningfully, demonstrably lower. And in a threat landscape where the question is not whether attacks will be attempted but whether they will succeed, that reduction in probability is precisely what security investment is supposed to deliver.

6. The Three Layers of Security Maturity

The argument developed in the preceding sections is not that compliance is wrong, or that penetration testing is without value. It is that security assurance is not a single thing — it is a stack, and each layer of that stack answers a different question. The mistake that creates real risk is treating any one layer as a substitute for the others, or assuming that reaching a certain layer means the work is done.

It helps to make that stack explicit.

Layer 1 — Governance and Process. This is the foundation that compliance frameworks and certifications are designed to establish and verify. It answers the question: does this organization behave like a security-conscious company? Named responsibilities, documented policies, defined incident response procedures, regular training, external audit — these are the building blocks of organizational security maturity. Without this layer, nothing else holds. A technically excellent security testing program sitting on top of dysfunctional governance will not save an organization from the predictable failures that good process is designed to prevent. This layer is necessary, and demanding evidence of it from vendors is entirely appropriate.

Layer 2 — Point-in-time Technical Validation. This is the layer that penetration testing occupies. It answers the question: did a qualified team find exploitable vulnerabilities when they examined this product under defined conditions? It adds technical specificity that governance assessments cannot provide — real findings, real attack paths, real remediation guidance. For many compliance frameworks it is also a formal requirement, and it satisfies that requirement appropriately. The limitations discussed earlier do not negate its value; they define its scope. A penetration test is a useful and necessary input. It is not, on its own, a sufficient picture of the product's security posture.

Layer 3 — Continuous Adversarial Scrutiny. This is the layer that bug bounty programs occupy, and it is the layer that most organizations — and most vendors — have not yet fully established. It answers the question: is this product able to withstand the attention of skilled, motivated, specialized researchers operating without artificial constraints? It reflects the conditions under which real attacks occur. It runs continuously, adapts as the product evolves, and produces findings that the other two layers are structurally unable to surface. It does not replace Layers 1 and 2 — it depends on them. But without it, the security picture remains fundamentally incomplete.

Most organizations operating today have Layer 1 in place. Many have Layer 2 in some form. Relatively few have invested seriously in Layer 3 — and that gap is where the residual risk that leads to breaches tends to live.

The practical implication is straightforward. When evaluating a vendor's security posture, evidence of Layers 1 and 2 should be considered a baseline expectation, not a differentiator. The meaningful question — the one that separates vendors who are genuinely committed to security from those who are committed to the appearance of it — is whether Layer 3 is present, how seriously it is resourced, and how long it has been running.

A vendor who can answer that question with specificity and transparency is a vendor who has made a different kind of commitment. Not to passing an audit, but to continuously exposing their product to the scrutiny it will face in the real world — and making it better as a result.

7. What This Means for You as a Buyer

Security assurance is ultimately a supply chain question. The policies your organization maintains, the controls you have implemented, the certifications you hold — none of these fully protect you from a vulnerability in a product you depend on. Your vendor's security posture is part of your security posture, whether your procurement process reflects that or not.

This is the practical implication of everything discussed so far, and it points toward a concrete change in how vendor security should be evaluated.

Asking for a penetration test report or a compliance certificate should remain part of due diligence — those documents tell you something real, and a vendor who cannot produce them is a vendor who has not cleared the baseline. But stopping there means accepting a security evaluation that ends precisely where the most important question begins.

The additional questions worth asking — and the answers worth paying attention to — are these:

Do you run a bug bounty program? A yes or no answer is informative on its own. A vendor with a running program has made a public commitment to ongoing scrutiny. A vendor without one has not.

Is the program public or private? Public programs invite scrutiny from any qualified researcher. Private programs restrict participation to an invited group. Both have legitimate uses, but a public program represents a higher level of openness and confidence in the product.

How long has the program been running? Longevity matters. A program that has been running for several years has produced compounding security improvements that a recently launched program has not yet had time to deliver. The age of the program is a proxy for the depth of scrutiny the product has received.

What does the scope cover? A program whose scope excludes large portions of the product's attack surface offers weaker assurance than one that covers the product comprehensively. Understanding what is and is not in scope tells you where the vendor's confidence — and caution — actually lies.

Are the bounty rewards meaningful? A program exists on paper even if its reward levels are set too low to attract serious researchers. Asking about payout ranges — particularly for high and critical severity findings — gives you a signal about how genuinely the vendor values external scrutiny. A vendor who has invested in competitive rewards has made a deliberate choice to attract skilled researchers. A vendor whose rewards are nominal may be running a program designed more for appearance than for outcomes.

How does the program feed into your development process? The value of a bug bounty program is not just in the findings — it is in what happens to them. A vendor who can describe a clear, fast path from finding to remediation to deployment is one whose program is genuinely integrated into their security practice, not maintained as a marketing signal.

A vendor who can answer these questions with specificity and transparency is demonstrating something that no certificate can convey: that they have chosen to hold their product to a standard defined not by auditors, but by the same adversarial conditions their customers face every day.

That is a meaningful differentiator. And in a landscape where breaches continue to occur at organizations with full compliance portfolios, it may be the most important security question a buyer can ask.

8. Kiteworks' Commitment to Continuous Security

At Kiteworks, the decision to invest seriously in bug bounty programs was not a response to an audit requirement or a customer request. It was a recognition that the standards described in this paper — the ones we are asking you to hold your vendors to — need to apply to us first.

Our investment in bug bounty is not a single program but a comprehensive portfolio. For the core Kiteworks product, we run multiple programs — both private and public — across two independent platforms. This structure is deliberate: private programs allow us to work with a trusted group of researchers on sensitive or complex areas of the product, while our public program opens the core attack surface to scrutiny from the broader research community. Running programs in parallel on separate platforms further diversifies the researcher pool and reduces the risk of blind spots that any single platform or community might carry.

That commitment extends beyond the core product. Every subsidiary product within the Kiteworks group runs its own bug bounty program — a standard we apply to all products upon acquisition. For customers of companies that become part of

the Kiteworks family, this represents a meaningful and immediate uplift: in virtually every case, the products we have acquired did not have a bug bounty program in place prior to joining the group. Establishing one is among the first security investments we make. It is a concrete expression of how seriously Kiteworks takes its responsibility to the customers it inherits, and a signal that acquisition into the Kiteworks group means moving toward a higher standard of security scrutiny, not away from it.

Each program is resourced with reward levels designed to attract and retain researchers with the skills and motivation to find what matters. We treat findings not as isolated reports to be closed but as operational inputs: discoveries drive remediation, remediation drives architectural improvement, and the cumulative result is a product portfolio that has been continuously tested against adversarial conditions rather than periodically reviewed against a checklist.

We are not in a position to guarantee that no vulnerability will ever be found in our products — no serious security program makes that claim, and any vendor who does should be viewed with skepticism. What we can say is that we have built the structure most likely to find vulnerabilities before an attacker does, and to respond to them quickly and transparently when they are identified.

For our customers, that commitment translates into something concrete: products that are continuously scrutinized, continuously improved, and held to a standard defined not by what compliance requires but by what genuine security demands.

9. Conclusion

The security landscape has matured significantly over the past decade. Compliance frameworks have become more rigorous, penetration testing has become standard practice, and organizational awareness of security risk has never been higher. That progress is real and worth acknowledging.

But the threat landscape has matured alongside it. Attackers are more patient, more specialized, and more creative than the testing models most organizations rely on were designed to anticipate. The gap between what compliance delivers and what genuine security requires has not closed — in many respects, it has widened.

The argument this paper has made is not a complicated one. Governance and certification establish that an organization takes security seriously. Penetration testing provides a bounded, time-limited view of a product's vulnerability surface. Bug bounty programs provide what neither of those can: continuous, adversarially realistic scrutiny from a global pool of specialized researchers, incentivized to find what others miss and given the time to do so.

Each layer matters. None is a substitute for the others. But the layer that most directly determines whether a sophisticated attacker will find an exploitable vulnerability before your vendor does — that layer is the one that most vendor security conversations still fail to reach.

For security leaders evaluating vendors, the message is straightforward: hold the baseline, but raise the bar. Certifications and penetration test reports are the beginning of a responsible security conversation, not the end of it. The vendors who are genuinely ahead of the threat are the ones who have gone further — who have opened their products to continuous scrutiny, invested in making that scrutiny meaningful, and built the internal processes to act on what it reveals.

When the question eventually asked is not whether a certificate was in place but whether everything possible was done to prevent a breach — and it will be asked — the answer that matters is whether your vendor had all three layers in place. Not just the ones that satisfy an auditor. All of them.