Maturity Models vs. Traditional Standards in Application Security

by Dinis Cruz and ChatGPT Deep Research, 2025/04/02

Introduction: In cybersecurity, organizations often struggle to measure and improve their application security. Traditional security standards and policies offer checklists of controls, but they can be rigid and one-size-fits-all. Maturity models like the OWASP Software Assurance Maturity Model (SAMM) and the Building Security In Maturity Model (BSIMM) provide a more flexible roadmap. They reflect how security practices can evolve over time rather than just meeting a static list of requirements. This paper explores the value of maturity models compared to traditional standards, using OWASP SAMM, BSIMM, and OWASP's Application Security Verification Standard (ASVS) as examples. We'll see how maturity models capture real-world variations in security, support risk-based decisions, scale controls to application criticality, promote continuous improvement, and adapt to different teams and products. Practical examples – such as using maturity levels to gate data access between systems – will illustrate how mapping current practices to maturity levels drives pragmatic security improvements. We'll also discuss how maturity models can feed into security scorecards or labels to transparently communicate an application's security posture.

Real-World Variation in Security Maturity

Every organization's security capabilities are different. Traditional standards often assume a uniform set of controls for all teams or applications, which may not reflect reality. Maturity models embrace these variations by defining levels or stages of security practice. For example, OWASP SAMM defines three maturity levels (with an implicit starting point of 0) for each security practice. Importantly, SAMM does **not** insist that every organization achieve the highest level in every practice. Instead, it recognizes that appropriate maturity depends on an organization's resources, risk environment, and mission. This flexibility mirrors the real world: a small startup and a large bank have different security maturity goals.

BSIMM, on the other hand, is descriptive – it observes what many organizations are doing. BSIMM provides a snapshot of common security activities across 100+ firms and offers an industry benchmark. However, simply copying industry practices may not yield the best ROI for *your* organization.

Many BSIMM participant companies are large enterprises, so it's unclear if those activities fit small or medium businesses. Maturity models like SAMM can scale down or up; practitioners note that SAMM is suitable for any size organization, including those smaller firms where BSIMM's one-size approach might falter. In essence, maturity models capture each organization's unique journey – including its culture, processes, and constraints – rather than enforcing identical standards across the board.

Risk-Based Decision Making with Maturity Models

A major advantage of maturity models is how they support risk-based decision making. Instead of blindly implementing controls, organizations can prioritize security improvements based on risk. OWASP SAMM explicitly encourages planning your security roadmap according to your risk profile, driven by metrics. This means teams assess which areas (like secure coding, testing, or incident response) pose the greatest risk and focus maturity efforts there first. Traditional standards might list many requirements without guidance on which are most critical, potentially leading to wasted effort on low-risk areas.

Maturity models help avoid that by tying progress to risk reduction. For example, SAMM's **Design** practices include **Threat Assessment**, where one activity is *application risk profiling*. This helps organizations identify which applications would cause the most damage if compromised. A team might find that a customer-facing portal carries higher risk than an internal tool; accordingly, they can prioritize maturing the security practices around the portal (e.g. stronger architecture review or more frequent testing). By mapping current practices to a maturity level, gaps become apparent in a risk context – maybe the portal is only at maturity level 1 in **Verification** (basic testing) when it really should be level 3 for such a high-risk app. Management can use this insight to drive targeted improvements that meaningfully reduce risk, rather than blanket policies.

Risk-based maturity also enables smarter decisions like "gating" – restricting certain actions until maturity criteria are met. For instance, an organization may decide that any application handling sensitive customer data must reach a higher maturity level (say, SAMM level 2 in **Implementation** and **Operations**) before it's allowed to go live or connect to production data. This gate ensures critical data isn't entrusted to applications with only ad-hoc security processes. Another example: If a development team at maturity level 1 (initial stage) wants to integrate their system with a high-value payment system, security leadership could require them to first advance to maturity level 2 in **Secure Architecture** and **Identity & Access Management**. Such gating, guided by maturity assessments, uses risk as the filter – high-risk integrations demand higher maturity. These practical checkpoints are difficult to establish with traditional compliance checklists, but maturity models provide a structured way to implement them, balancing security with business needs.

Scaling Security Controls to Application Criticality

Not all applications need the same level of security rigor. Maturity models allow organizations to scale controls relative to each application's criticality. A low-risk brochure website shouldn't have the exact same costly security controls as a high-risk financial transaction system. OWASP's Application Security Verification Standard (ASVS) illustrates this principle well: it defines three levels of security assurance to match different application needs. **Level 1** is for low-risk applications with minimal security requirements; **Level 2** is for most business applications needing comprehensive controls; **Level 3** is reserved for highly sensitive apps (like those handling financial or healthcare data) where advanced security is essential. An organization can choose an ASVS level based on the criticality of each application – for example, internal tools might only target ASVS Level 1, while customer-facing apps with sensitive data aim for Level 2 or 3. This tiered approach ensures effort scales with importance.

Maturity models like SAMM and BSIMM complement this by providing a broader organizational view. You might use ASVS (a standard) to verify that a given application meets the needed technical controls at a certain level, while using SAMM to ensure the **process** and **practice maturity** around that application also scale appropriately. For instance, a safety-critical system not only should meet ASVS Level 3 technical requirements, but the team should also be at higher maturity in SAMM's **Governance** and **Operations** functions (meaning strong security policy compliance, metrics, and incident response are in place). Together, this ensures that both the code and the surrounding practices are mature enough for the application's criticality. In practice, this might mean assigning more resources (like security architects or specialized testing) to high-criticality projects to help them reach higher maturity levels, while lower-criticality projects get a "lighter" touch. This risk-based scaling of controls is far more efficient than a uniform standard for all, and it helps organizations invest security effort where it matters most.

Continuous Evolution and Improvement Over Time

Unlike a static policy document, maturity models are built around the idea of continuous improvement. They encourage organizations to **evolve** their security posture iteratively. OWASP SAMM explicitly highlights this benefit – it helps build a balanced security assurance program in well-defined iterations and demonstrate concrete improvements over time. Each maturity level serves as a milestone: reaching level 1 might mean you have basic policies and some training in place; level 2 might add deeper practices like threat modeling and code review; level 3 might indicate a fully ingrained security culture with automation and metrics. This progression motivates teams to keep improving, and it provides a sense of achievement at each step.

Traditional standards often focus on a point-in-time compliance ("are you doing X, yes or no?"). They might not clearly show how to progress beyond mere compliance. Maturity models fill that gap by being a roadmap. For example, SAMM's maturity streams provide prescriptive guidance on how to move from one level to the next. If at maturity 1 you perform security testing occasionally, the model might prescribe at maturity 2 to integrate testing into CI/CD pipelines, and at level 3 to include advanced techniques like fuzzing or manual code reviews by experts. This staged approach fosters a culture of **continuous improvement** – security isn't a one-and-done task but an ongoing journey.

Another advantage is that maturity models allow measuring improvement. By assessing your SAMM level every quarter or year, you can concretely see progress (e.g., "Last year our **Verification** practices were at 1, now they're 2 – we added automated testing and threat modeling"). This measurement can be more meaningful to the business than a pass/fail audit. It shows direction and momentum. And if some area's maturity plateaus, it flags a potential issue to investigate or a decision to be made (maybe that's okay due to business context, or maybe it needs management support to push further). Ultimately, maturity models turn security into a continuous process of evolution – encouraging organizations to adapt and improve as new threats emerge or as the business grows.

Customization for Teams, Products, or Organizations

Security maturity models are not only about levels – they're about flexibility in implementation. Different teams and products have different development styles and constraints. A key strength of models like SAMM is that they can be tailored to these differences. As one practitioner noted, *"SAMM tends to be a little less piecemeal than ASVS. It's very general, and it's built that way so that it is flexible and [so] different teams with different development styles can use it."* (Using OWASP's SAMM and ASVS Together | Pivot Point). In practice, this means an organization can customize how it implements each practice. For example, one product team might fulfill the **Education & Guidance** practice by conducting regular in-person secure coding workshops (to reach SAMM level 2 in that stream), while another team in the same company might use an online training platform and monthly security quizzes to achieve the same objective. The model provides the *what* (e.g., "ensure developers are trained") but teams can choose the *how* that best fits their workflow.

Customization also applies at the organizational level. A fintech startup might focus first on maturity in areas like **Threat Assessment** and **Secure Deployment** which are critical to protecting customer transactions, whereas a healthcare software company might prioritize **Privacy** and **Data Security** practices. Maturity models aren't a rigid prescription; they allow you to pick and choose focus areas based on your context. SAMM explicitly supports this by being open and modular – you can even start with a small scope. In fact, because SAMM is open-source and community-driven, it's

feasible to experiment on a subset of the organization or a pilot project. This lets teams learn and adapt the model to their reality. Contrast this with some traditional standards that might require full organization-wide adoption to be effective, or proprietary models like BSIMM where you have less freedom to adapt the framework itself.

Another benefit of customization is that maturity models can integrate with existing processes. If a team is Agile, there are guidelines for applying SAMM in Agile environments. If a company already follows ISO or NIST standards, a maturity model can map onto those requirements and show progress beyond the minimum. For example, a team complying with OWASP ASVS (which is a standard) can use SAMM to gauge how well that compliance is embedded in their development lifecycle and culture. Over time, the organization might even extend the model – creating a custom maturity model by adding domains or practices unique to its business (since SAMM is extensible). In summary, maturity models promote a *"choose your own adventure"* approach to security improvement: they set a direction and milestones but let each team or product find the path that works best for them.

Maturity Models for Scorecards and Security Labels

Communicating security posture is often challenging – especially to non-technical stakeholders. This is where maturity models can underpin **cybersecurity scorecards or labels** to provide a clear, adaptable way of showing how secure an application or system is. Imagine an internal scorecard where each critical application is rated, say, **Security Maturity: Level 2 (of 3)**, along with a few key indicators (like which practices are at what level). This gives management a high-level snapshot. It's similar to a report card for security – much more nuanced than simply "compliant or not". Because maturity levels are well-defined, they offer transparency. For instance, a scorecard might show that an e-commerce platform is at maturity level 3 in **Operations** (meaning strong incident response and monitoring), but only level 1 in **Design** (perhaps lacking threat modeling). Stakeholders can immediately grasp where improvements are needed and where the system is strong.

Such scorecards can be used both internally and externally. Internally, they help allocate resources: if one app's security maturity is lagging behind, it might need more attention or budget. Externally, some organizations might share maturity information with partners or customers as a trust signal. We see an analogy in government and industry initiatives like the Cybersecurity Maturity Model Certification (CMMC), which essentially **labels** suppliers with a maturity level that they must achieve to handle certain data. While CMMC is aimed at suppliers, a similar concept could apply to applications or products: e.g., a SaaS product could publish that it meets "AppSec Maturity Level 2" based on OWASP SAMM assessments, giving clients confidence in its security processes. There are ongoing efforts in the industry to create consumer-friendly security labels (especially for IoT

devices and consumer software) ([PDF] Consumer Cybersecurity Labeling for IoT Products: Discussion Draft ...). Maturity models can feed into these by providing a quantifiable evaluation that is more dynamic than a yes/no compliance checklist.

A key benefit of using maturity-based labels is adaptability. Security isn't static – new threats emerge and systems change. A label or scorecard tied to a maturity model can be updated as an organization improves (or if, unfortunately, it regresses). It avoids giving a false sense of security; instead of saying "Product X is Secure" (which might go out of date), the label might say "Product X Security Maturity: 2/3, assessed Q1 2025". If next year the product advances to level 3 by adding more controls and process improvements, the label can reflect that. This transparency creates an incentive for continual improvement, as customers and leadership can track progress. It also allows flexibility: if an application cannot feasibly reach the highest level due to its context, that nuance can be communicated (maybe it's a simple app that doesn't need level 3, and stakeholders understand that from the label description). Overall, maturity models provide a structured language to describe security posture, which makes scorecards and labels much more meaningful and trustable for everyone involved.

Conclusion

Maturity models like OWASP SAMM and BSIMM offer a practical complement (or alternative) to traditional security standards and policies. Where standards provide lists of what to do, maturity models show **how well you're doing** and how to improve. They capture real-world differences in security maturity across teams and applications, letting organizations shape their security program to fit their unique risk landscape. By focusing on progression and continuous improvement, these models support risk-based decisions – ensuring effort is spent where it reduces risk the most – and enable scaling of security controls according to each application's criticality. Unlike rigid checklists, maturity models are living frameworks: they encourage evolution over time and can be customized to different team structures, development methodologies, and business needs.

Practical use cases demonstrate their value. Organizations can use maturity levels as gates or quality checks – for example, only allowing high-risk data or system access to applications that have reached a certain maturity. Mapping current practices to a model highlights gaps and guides **pragmatic security improvements** rather than theoretical ones. Finally, by incorporating maturity assessments into scorecards or security labels, organizations gain a transparent way to communicate security posture. This not only helps internal governance and tracking but can build trust with users and partners by showing an honest, up-to-date picture of security efforts. In today's fast-evolving threat environment, having that adaptive, improvement-focused approach is invaluable. Maturity models don't replace the fundamentals of security – they enhance them by providing context and direction, ensuring that security programs remain effective, scalable, and aligned with real-world needs over time.