OWASP SAMM: Patched, Tweaked, and Scaled

Michael Markevich
Cybersecurity Expert

Moldova DevCon | Chișinău, October 9-11, 2025

Who Am I

- Cybersecurity practitioner with 25+ years of experience
- 3 x CISO, startup advisor and founder
- Career mentor and academic lecturer
- Open-source developer and advocate

Why This Talk

Thanks to AI, now we can find more security bugs in software than ever before.

But what happens with the security of the product over time?

Our Software Is Secure

What does it actually mean? *And when?*

Transparency and Trust

Customers trust us to deliver a secure product.

But how can we support this trust claim?

What Would You Do?

Nothing! Our security quality is good enough.

What Would You Do?

Use SAST, DAST, threat modelling. Just because others do the same.

What Would You Do?

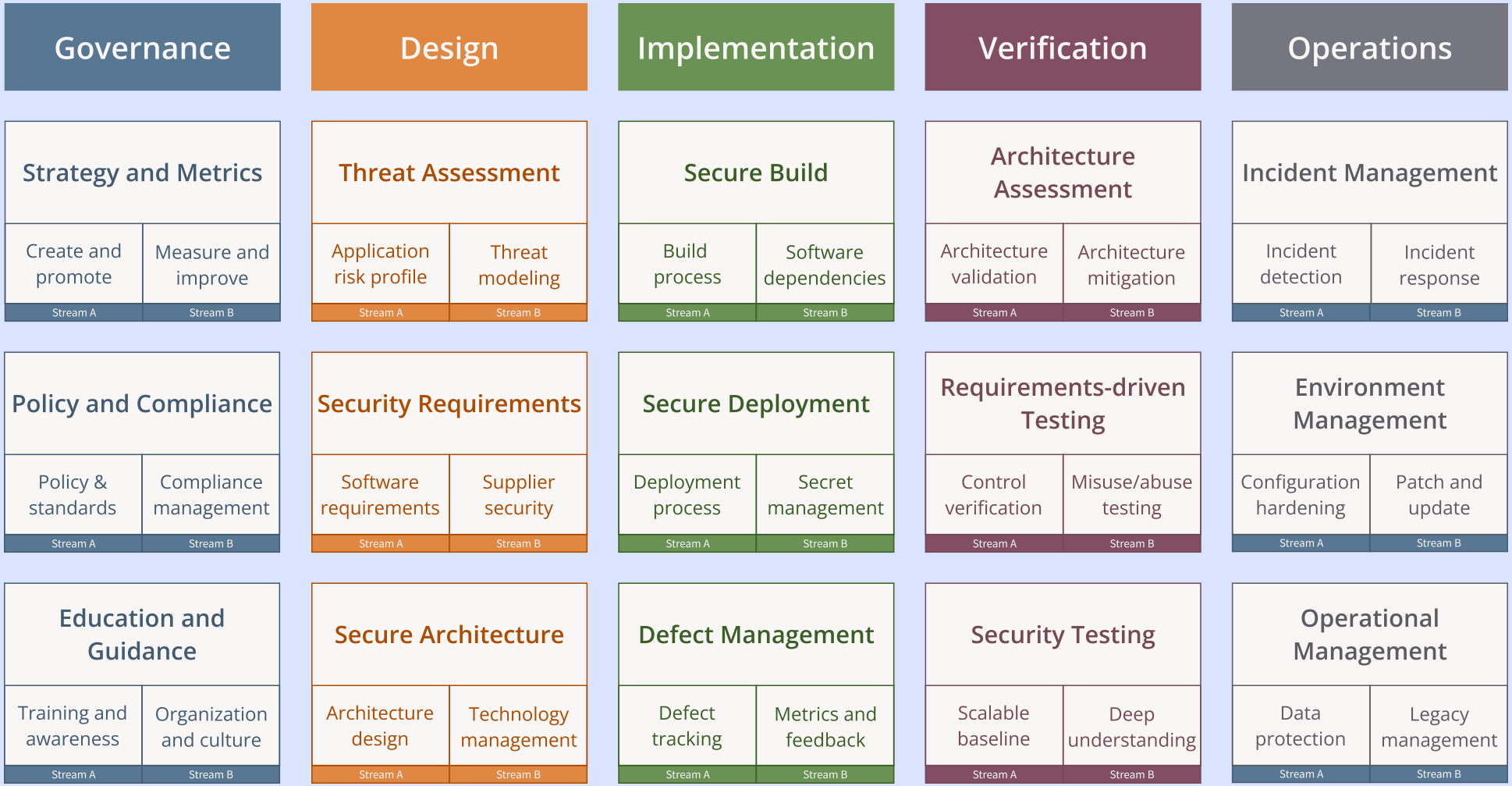
Use a systematic approach. Sounds boring, but hold on.

Systematic Approach To Security Maturity

Two most popular frameworks to measure and improve software security maturity: BSIMM (commercial) and OWASP SAMM (open-source).

- SAMM (prescriptive): Here’s how to do it
- BSIMM (descriptive): Here’s what others are actually doing

Inside SAMM


Governance
Design
Implementation
Verification
Operations
Strategy and Metrics
Create and promote
Measure and improve
Stream A
Stream B
Threat Assessment
Application risk profile
Threat modeling
Stream A
Stream B
Secure Build
Build process
Software dependencies
Stream A
Stream B
Architecture Assessment
Architecture validation
Architecture mitigation
Stream A
Stream B
Incident Management
Incident detection
Incident response
Stream A
Stream B
Policy and Compliance
Policy & standards
Compliance management
Stream A
Stream B
Security Requirements
Software requirements
Supplier security
Stream A
Stream B
Secure Deployment
Deployment process
Secret management
Stream A
Stream B
Requirements-driven Testing
Control verification
Misuse/abuse testing
Stream A
Stream B
Environment Management
Configuration hardening
Patch and update
Stream A
Stream B
Education and Guidance
Training and awareness
Organization and culture
Stream A
Stream B
Secure Architecture
Architecture design
Technology management
Stream A
Stream B
Defect Management
Defect tracking
Metrics and feedback
Stream A
Stream B
Security Testing
Scalable baseline
Deep understanding
Stream A
Stream B
Operational Management
Data protection
Legacy management
Stream A
Stream B

How To Collect Input

- Interviews
- Self-assessments

Questionnaire Example

Secret Management	**1**	**Do you limit access to application secrets according to the least privilege principle?**	**A**	**Yes, for most or all of the applications**
		You store production secrets protected in a secured location Developers do not have access to production secrets Production secrets are not available in non-production environments		
	2	**Do you inject production secrets into configuration files during deployment?**	**A**	**Yes, for some applications**
		Source code files no longer contain active application secrets Under normal circumstances, no humans access secrets during deployment procedures You log and alert when abnormal secrets access is attempted		
	3	**Do you practice proper lifecycle management for application secrets?**	**A**	**No**
		You generate and synchronize secrets using a vetted solution Secrets are different between different application instances Secrets are regularly updated		
Defect Management				**Answer**
Defect Tracking	**1**	**Do you track all known security defects in accessible locations?**	**A**	**Yes, for at least half of the**
		You can easily get an overview of all security defects impacting one application You have at least a rudimentary classification scheme in place The process includes a strategy for handling false positives and duplicate entries The defect management system covers defects from various sources and activities		
	2	**Do you keep an overview of the state of security defects across the organization?**	**A**	**Yes, for at least half of the applications**
		A single severity scheme is applied to all defects across the organization		

The Setting

- A technology company with 20+ autonomous product teams
- Mix of web, mobile, backend apps
- Shared security approach but independent delivery cycles

The Goal

Measure security maturity of all teams and the company overall, and understand what to improve.

What Didn't Work

- One questionnaire for the whole company ignored team context
- Language was too abstract
- The questionnaire was too big
- Some teams felt it was irrelevant or added no value

What Finally Worked

- One questionnaire per product team, completed annually
- Prefilled common items
- Simplified and more specific questions
- Product owners and other stakeholders involved

Scoring

- Per product team and global score
- Annual comparison between teams
- Areas for improvement

Scoring Example

Current Maturity Score					
			Maturity		
Business Functions	Security Practices	Score	1	2	3
Governance	Strategy & Metrics	0.25	0.00	0.13	0.13
Governance	Policy & Compliance	0.50	0.50	0.00	0.00
Governance	Education & Guidance	1.00	0.50	0.00	0.50
Design	Threat Assessment	0.00	0.00	0.00	0.00
Design	Security Requirements	0.25	0.13	0.13	0.00
Design	Secure Architecture	0.25	0.13	0.13	0.00
Implementation	Secure Build	0.38	0.25	0.13	0.00
Implementation	Secure Deployment	1.25	0.75	0.38	0.13
Implementation	Defect Management	1.00	0.50	0.50	0.00
Verification	Architecture Assessment	0.25	0.25	0.00	0.00

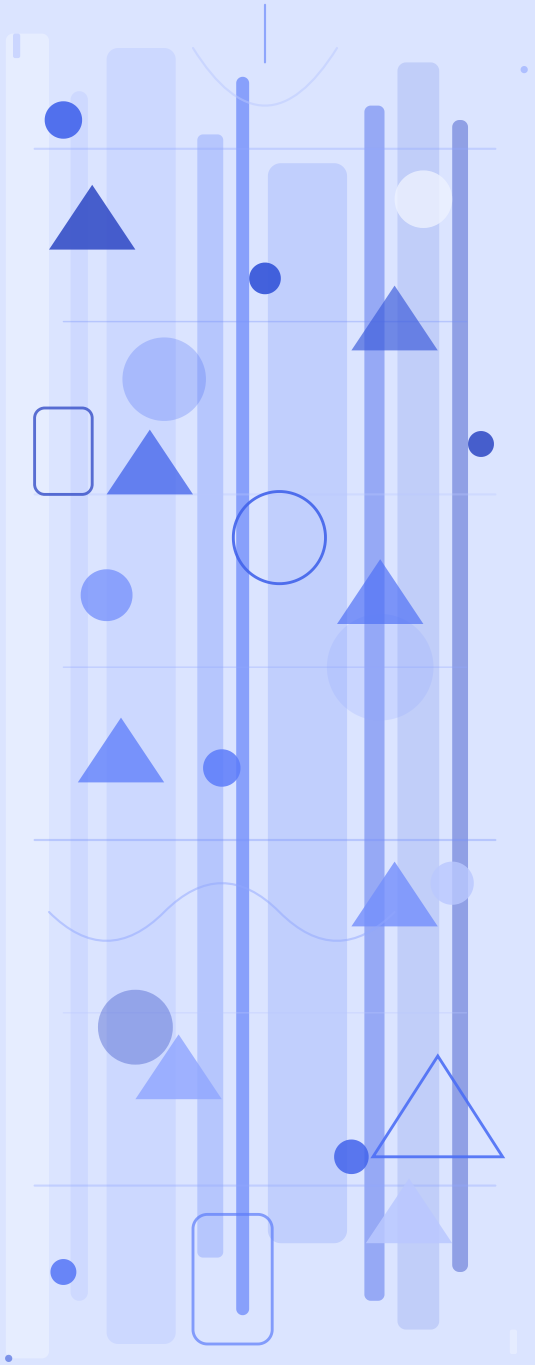
Business Functions	Score
Governance	0.58
Design	0.17
Implementation	0.88
Verification	0.33
Operations	0.25
Overall	0.44

Outcomes

- Increased security awareness in the teams
- Meaningful maturity metrics that evolve over time
- Evidence of security maturity for customers

Final Notes

- SAMM provides a comprehensive view on security maturity
- No need to implement all the requirements, but consider the risks
- Useful for roadmap planning



Thank You