Realizing Software Security Maturity

Matthew Fanto
Duo Security
Low Security Maturity

- **Staffing:** Non-existent — “Larry used Metasploit once, I think?”
- **Self Awareness:** Poor — “Here’s our ‘Hacker Proof’ web site badge.”
- **Coverage:** Unknown — “All of the software we can find seems OK?”
- **Focus:** Staying afloat — “Our PCI ASV scan came back fine!”
- **Metrics:** Hand-wavy — “We didn’t have any SQL data exfiltrated in Q4!”
Medium Security Maturity

- **Staffing**: Hiring security-minded engineers who do the best they can
- **Self Awareness**: Confident in key tactical areas, but not the “big picture”
- **Coverage**: Annual assessment by a 3rd-party firm of major code bases
- **Focus**: Meeting compliance needs & tunnel vision on the OWASP Top 10
- **Metrics**: Reducing defects, indexed on issue count instead of real-world risk
High Security Maturity

- **Staffing:** Has a dedicated team focused on application security initiatives
- **Self Awareness:** Understands scope, real-world threats, and program gaps
- **Coverage:** Security activities applied to projects throughout the SDLC
- **Focus:** Building scalable processes, testing methodologies, and consistency
- **Metrics:** Maturity model-based prioritization & coverage of AppSec functions
Common Roles for an Application Security Team

- **Application Security Analyst**: Handles inbound security defect verification, root cause analysis, resolution task creation, and ongoing bug management.
- **Application Security Engineer**: Performs security activities, including: design reviews; threat models; code auditing; and security assessments.
- **Security Architect**: Focuses on defining the security properties of software specifications, deployment architecture, and implementation requirements.
- **Governance & Compliance Lead**: Manages the maturity model, defines security standards, leads training initiatives, and supports compliance needs.
Tactical vs. Strategic Application Security
Without a Program:
- One-off “heroics” by engineers
- Inconsistent, best-effort coverage
- Unclear growth & maturity of AppSec
- Piecemeal, irregular assessment
- A lot of hammers for mostly screws

With a Program:
- Clear expectations for stakeholders
- Consistent and prioritized coverage
- An evident horizon of AppSec maturity
- Activities to support the entire SDLC
- Efficient, purpose-driven capabilities
Without a Program:
- One-off “heroics” by engineers
- *Inconsistent, best-effort coverage*
- Unclear growth & maturity of AppSec
- Piecemeal, irregular assessment
- A lot of hammers for mostly screws

With a Program:
- Clear expectations for stakeholders
- *Consistent and prioritized coverage*
- An evident horizon of AppSec maturity
- Activities to support the entire SDLC
- Efficient, purpose-driven capabilities
● Without a Program:
  ○ One-off “heroics” by engineers
  ○ Inconsistent, best-effort coverage
  ○ *Unclear growth & maturity of AppSec*
  ○ Piecemeal, irregular assessment
  ○ A lot of hammers for mostly screws

● With a Program:
  ○ Clear expectations for stakeholders
  ○ Consistent and prioritized coverage
  ○ *An evident horizon of AppSec maturity*
  ○ Activities to support the entire SDLC
  ○ Efficient, purpose-driven capabilities
● Without a Program:
  ○ One-off “heroics” by engineers
  ○ Inconsistent, best-effort coverage
  ○ Unclear growth & maturity of AppSec
  ○ *Piecemeal, irregular assessment*
  ○ A lot of hammers for mostly screws

● With a Program:
  ○ Clear expectations for stakeholders
  ○ Consistent and prioritized coverage
  ○ An evident horizon of AppSec maturity
  ○ *Activities to support the entire SDLC*
  ○ Efficient, purpose-driven capabilities
Without a Program:
- One-off “heroics” by engineers
- Inconsistent, best-effort coverage
- Unclear growth & maturity of AppSec
- Piecemeal, irregular assessment
- A lot of hammers for mostly screws

With a Program:
- Clear expectations for stakeholders
- Consistent and prioritized coverage
- An evident horizon of AppSec maturity
- Activities to support the entire SDLC
- Efficient, purpose-driven capabilities
Metrics: Without a Program

“How many bugs did we find this time versus last time?”

“I Found a Lot of Bugs!”
You’re probably only doing just-in-time security assessments of code bases.

“I Found No Bugs!”
Your security assessment capabilities could be incomplete or too tightly-scoped.

Building an Application Security program allows you to have many signals of how success is measured... bug counting should not be the basis of “good” vs. “bad.”
Metrics: With a Program

- “What portion of code bases have had formal security assessment?”
- “What percentage of languages have secure coding standards?”
- “How often do engineers receive role-focused security training?”
- “What is the mean time to ‘security defect’ resolution?”
- “What percentage of code bases have security integration tests?”
A Model for Maturity
## BSIMM & SAMM: A Comparison(ish)

<table>
<thead>
<tr>
<th></th>
<th>BSIMM</th>
<th>SAMM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
<td>Building Security in Maturity Model</td>
<td>Software Assurance Maturity Model</td>
</tr>
<tr>
<td><strong>In Use Since</strong></td>
<td>2008</td>
<td>2009 (1.0)</td>
</tr>
<tr>
<td><strong>Latest Release</strong></td>
<td>8 (September 2017)</td>
<td>1.5 (April 2017)</td>
</tr>
<tr>
<td><strong>Curated By</strong></td>
<td>Synopsys (Security Vendor)</td>
<td>OWASP (Community Organization)</td>
</tr>
<tr>
<td><strong>Model Basis</strong></td>
<td>Real-world, “in use” industry data</td>
<td>“Ideal state” via input from the community</td>
</tr>
<tr>
<td><strong># of Top-Level</strong></td>
<td>4 — Governance, Intelligence, SSDL Touchpoints, and Deployment</td>
<td>4 — Governance, Construction, Verification, and Operations</td>
</tr>
<tr>
<td><strong>Groupings</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong># of Activities</strong></td>
<td>113 across 12 sub-groupings</td>
<td>77 across 12 sub-groupings</td>
</tr>
</tbody>
</table>
[SE2.4: 29] Use code signing.
The organization uses code signing for software published across trust boundaries. Code signing is particularly useful for protecting the integrity of software that leaves the organization’s control, such as shrink-wrapped applications or thick clients. The fact that some mobile platforms require application code to be signed does not indicate institutional use of code signing.

B. Perform code signing for application components

Though often used with special-purpose software, code signing allows users and operators to perform integrity checks on software such that they can cryptographically verify the authenticity of a module or release. By signing software modules, the project team enables deployments to operate with a greater degree of assurance against any corruption or modification of the deployed software in its operating environment.

Signing code incurs overhead for management of signing credentials for the organization. An organization must follow safe key management processes to ensure the ongoing confidentiality of the signing keys. When dealing with any cryptographic keys, project stakeholders must also consider plans for dealing with common operational problems related to cryptography such as key rotation, key compromise, or key loss.

Since code signing is not appropriate for everything, architects and developers should work with security auditors and business stakeholders to determine which parts of the software should be signed. As projects evolve, this list should be reviewed with each release, especially when adding new modules or making changes to previously signed components.
Better Understanding BSIMM & SAMM

- BSIMM considers how organizations actually build an application security program, while SAMM considers how we think we should build a program.

- BSIMM provides real-world data that can be useful to compare & contrast your own organization against, including details about team staffing.

- SAMM enables community dialog around the horizon we should aspire to.

- These maturity models should *not* be treated as immutable checklists, but rather act as sources of influence and alignment for a program’s build out.
Our Application Security Team
Our Security Organization

Corporate Security  Data Science  Trust and Compliance

Security Research  Application Security  Product R&D

~10% of Our Employees are in Security Organization Roles
1.6% Percentage of Software Security Group (SSG) Members to Software Engineers in BSIMM8’s Data Set

8.3% Percentage of Our Application Security Team Members to Our Software Engineering Staff
8.3% Percentage of Security Software Group (SSG) Members to Software Engineers in BSIMM8’s Data Set

1.6% Percentage of Our Application Security Team Members to Our Software Engineering Staff
Application Security
Team Values
Engineering is Family

Application Security will be adversarial in activity, but never in the relationship with our Engineering team members.

What this means:
- Empathetic and respectful engagement
- Empower engineers with knowledge
- Be available, be thoughtful, be patient

What this does not mean:
Low Friction, High Value

Application Security will look for key points in the SDLC that provide high value, with low friction, to increase security.

What this means:
- Less roadblocks, more roundabouts
- Be mindful of overhead on Engineers
- Be creative in building better security

What this does not mean:
Build a Paved Road

Application Security will build and promote standard capabilities that accelerate engineers with clear support & benefits.

What this means:
- Guardrails so engineers feel confident
- Help to accelerate innovation & output
- More time to spend on “hard” problems

What this does not mean:
How Could it Go Right?

Application Security will ensure Engineering is enabled & supported to lead innovation, even for hard security challenges.

What this means:

- We’re enablers, not the team of “No”
- Our titles contain ‘Engineer’ for a reason
- Be up for the challenge; no fatalists here

What this does not mean:
No Code Left Behind

Application Security is committed to ensuring that no code is forgotten about and that our security testing accounts for it.

What this means:
- Don’t just focus on the new & shiny
- Understand the *full* software inventory
- “Old” code changes in “new” deploys

What this does not mean:
Our Application Security Program
Duo Application Security Maturity Model (DASMM)

**Governance**
- Strategy & Metrics
- Policy & Compliance
- Education & Guidance

**Engineering**
- Software Requirements
- Software Architecture
- Threat Assessment

**Verification**
- Code Review
- Software Testing
- Design Review

**Operations**
- Defect Management
- Deployment Composition

- 54 Activities
- 46 Activities
- 55 Activities
- 35 Activities

Leveraging Industry Maturity Models with the Ability to Customize
# DASMM: Tracking Program Maturity

<table>
<thead>
<tr>
<th>Coverage</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Consistent coverage and very mature practices</td>
</tr>
<tr>
<td>0.5</td>
<td>Inconsistent coverage and/or partially mature practices</td>
</tr>
<tr>
<td>0.2</td>
<td>Minor coverage and/or weak practices</td>
</tr>
<tr>
<td>0</td>
<td>Non-existent coverage and/or immature practices</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Priority</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>An activity vital to the success of the AppSec program</td>
</tr>
<tr>
<td>2</td>
<td>Highly valuable activities that notably increase maturity</td>
</tr>
<tr>
<td>3</td>
<td>Supplemental to program goals, but not key to success</td>
</tr>
<tr>
<td>4</td>
<td>There is no intention to adopt this activity in the future</td>
</tr>
</tbody>
</table>

* Spoiler Alert: Fake Data*
DASMM Is...

- An evolving view of our AppSec program
- Helpful to visualize what we *could* do
- Able to simplify compliance/audit needs
- A canonical way to reference activities
- Categorized groupings of our program

DASMM Is Not...

- A list of “things we have to do”
- Immutable in prioritization of activities
- Rigid to future additions/subtractions
- A checklist to define success or failure
- Something Engineers have to fear
DASMM in Practice

- Abstracted layer atop of BSIMM & SAMM to give a single view of multiple maturity models with some customization
- Cleaner mapping of our compliance teams’ various needs with the security initiatives that are underway or could be
- Direct mapping of team goals each quarter to how we want to “move the needle” of our maturity & coverage
Foundational
OWASP SAMM
Synopsys BSIMM
Microsoft SDL

Descriptive
Bugcrowd VRT
Microsoft STRIDE
Microsoft DREAD

Functional
FIRST PSIRT Framework
OWASP ASVS
ISO 30111 & 29147
## Strong Collaboration With Others

<table>
<thead>
<tr>
<th>Quality Assurance</th>
<th>Product Team</th>
<th>Compliance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximize testing coverage</td>
<td>Advise on industry trends</td>
<td>Vendor security assessments</td>
</tr>
<tr>
<td>Shared technical tooling</td>
<td>Assess early design risk</td>
<td>RFP questionnaire responses</td>
</tr>
<tr>
<td>Confirm/deny security bugs</td>
<td>Advocate “Security Quality”</td>
<td>Support auditor requirements</td>
</tr>
</tbody>
</table>
Give Back to the Community

**Content**
- Present at conferences
- Author blog posts
- Respond to press inquiries
- Publish white papers

**Industry Contributions**
- Influence relevant standards
- Build community events
- Perform security research
- Support public policy reform
Supporting Our #1 Customer

ENGINEERING
Security Development Lifecycle (SDL)

Training
- Requirements
- Design
- Implementation
- Verification
- Release
- Response
Security Development Lifecycle (SDL)

- **Engineering-focused “Security Skills & Interest” survey**
  - All new Engineering hires fill out this form to influence our program focus

- **Duo Engineering Vulnerability Discussion (DEVD)**
  - Short presentations on vulnerability classes and how they affect engineers

- **Hands-on formal training & guest speakers**
  - Tailored courses developed internally and 3rd-party specialized training

- **Informal gamified training**
  - Internal CTFs and Elevation of Privilege (EoP) card-game tournaments
Security Development Lifecycle (SDL)

Security Design Reviews

Evaluates the security architecture of an application's overall composition.

Benefits to Engineers

- Early, efficient clarity on secure design
- Reduces likelihood of major refactoring later
- Provides early AppSec team awareness
- Allows for highly interactive engagement

Possible Deliverables

- Real-time feedback
- Formalized review artifacts
- Software security requirements
### Security Development Lifecycle (SDL)

#### Threat Modeling

*Reviewing a software design to enumerate threats and contextualize their real risk.*

#### Benefits to Engineers

- Thoughtful evaluation of attack surface
- Development of a better “attacker mindset”
- Useful insights for cost/benefit analysis
- Allows for more strategic risk mitigation

#### Possible Deliverables

- Data flow diagrams
- Threat enumeration details
- Interactive whiteboarding
Security Development Lifecycle (SDL)

Training → Requirements → Design → Implementation → Verification → Release → Response

Code Auditing

*Point-in-time analysis of how implemented code has met the intent of security engineering principles, standards, and guidelines as defined for the project’s goals.*

Benefits to Engineers

- Prompt remediation of security anti-patterns
- Collaborative review of code in increments
- Focused attention to “security quality” of work
- Bite-sized security education opportunities

Possible Deliverables

- Well-documented remediation patches
- Detailed technical writeups of vulnerabilities
- Improved security test coverage
Security Development Lifecycle (SDL)

Security Assessment

Comprehensive review of software's total security composition, usually at major lifecycle inflection points (e.g. new release, feature update, major code refactor).

Benefits to Engineers

- Holistic review of entire in-scope code base
- Analyzes the integrated security properties
- New or updated view of threat model artifacts
- Good “gut check” before a major release

Possible Deliverables

- Threat modeling asset updates
- A comprehensive assessment report
- Detailed technical writeups of vulnerabilities
FIRST PSIRT Framework - Being finalized after a recent v1.0 RFC period, during which we submitted feedback. We plan to leverage this framework longer term.

Product Security Advisory (PSA) process
- Modeled after ISO/IEC 30111:2013
- All PSAs are archived on our web site after release

Coordinated vulnerability disclosure policy
- Modeled after ISO/IEC 29147:2014
- Our contact details are published on our web site, including a GPG key
Days From Vulnerability Report to PSA Release
How Do We Organize?
Our Engineering Support Workflow

- Engineering Needs Security Assistance
- Ad-hoc or Formalized?
  - Formalized
    - Intake
    - Execution
    - Defect Management
  - Ad-Hoc
    - Quick Meeting or Visit AppSec During Weekly Office Hours
- End Security Assistance
Ad-hoc Help (Easy Mode)

- Review small code diffs
- One-off Slack conversations
- Issue tracker subscriptions
- Forwarding us an email thread
- Walking up to our desk with beer
AppSec Team “Office Hours”

- 1 hour of weekly time with AppSec
  - Published on engineer calendars
  - Reminders via Slack & in-person

- Open-ended discussion and Q&A

- Often results in “next step” outcomes
  - Realizes low-friction, high-value

...and when we’re bored, we hack stuff ;)}
Formalized Help (Hard Mode)
Intake Process

1. Intake form is submitted by an engineer
2. AppSec team confirms receipt and reviews
3. Timeline and AppSec resources forecasted
4. Details added to the security activity board

The Intake Form Will Receive…

- Which activity was requested and why
- Overview of the request’s scope
- Major security concerns and focuses
- Platform & programming language details
- Links to all relevant project artifacts
- Activity timeline and point of contact
### Execution: 1st or 3rd Party?

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Application Security Team</th>
<th>Third-party Security Firm</th>
<th>Bug Bounty Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to identify and select experts who specialize in a given technical area</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Legal &amp; technical simplicity to share source code or privileged system access</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Diversity of expertise and perspective for a given assessment scoping</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Ease of supporting communications, triage, and remediation processes</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>[...]</td>
<td>[...]</td>
<td>[...]</td>
<td>[...]</td>
</tr>
</tbody>
</table>
### Execution Management and Scheduling

<table>
<thead>
<tr>
<th>Active</th>
<th>Activity</th>
<th>AppSec Lead</th>
<th>Requester</th>
<th>Kick Off</th>
<th>Work Started</th>
<th>Work Finished</th>
<th>Report Delivery</th>
<th>Debrief</th>
<th>Duration</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super Important Feature</td>
<td>Security Assessment</td>
<td>📊</td>
<td>📍</td>
<td>Oct 03</td>
<td>Oct 06</td>
<td>Oct 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Create a New Pulse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Activity</th>
<th>AppSec Lead</th>
<th>Requester</th>
<th>Kick Off</th>
<th>Work Started</th>
<th>Work Finished</th>
<th>Report Delivery</th>
<th>Debrief</th>
<th>Duration</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less important feature</td>
<td>Threat Model</td>
<td>📊</td>
<td>📍</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Create a New Pulse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Completed</th>
<th>Activity</th>
<th>AppSec Lead</th>
<th>Requester</th>
<th>Kick Off</th>
<th>Work Started</th>
<th>Work Finished</th>
<th>Report Delivery</th>
<th>Debrief</th>
<th>Duration</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Older Cool Feature</td>
<td>Threat Model</td>
<td>📊</td>
<td>📍</td>
<td>Sep 06</td>
<td>Sep 07</td>
<td>Sep 17</td>
<td>Sep 21</td>
<td>Sep 21</td>
<td>Sep 8 - 21</td>
<td><a href="https://www.f">https://www.f</a>...</td>
</tr>
<tr>
<td>Older Older Cool Feature</td>
<td>Security Design Review</td>
<td>📊</td>
<td>📍</td>
<td>Aug 14</td>
<td>Aug 14</td>
<td>Aug 25</td>
<td>Sep 01</td>
<td>Sep 01</td>
<td>Aug 14 - Sep 1</td>
<td><a href="https://www.f">https://www.f</a>...</td>
</tr>
<tr>
<td>Sketchy Feature</td>
<td>Code Auditing</td>
<td>📊</td>
<td>📍</td>
<td>Aug 01</td>
<td>Aug 03</td>
<td>Aug 09</td>
<td>Aug 14</td>
<td>Aug 16</td>
<td>Aug 1 - 16</td>
<td><a href="https://www.f">https://www.f</a>...</td>
</tr>
</tbody>
</table>

- Similar to an internal consultancy
- Easy and transparent scheduling
- Simple and repeatable process
- Helps answer statusing questions
1st Party Execution: Kick-Off Checklist

- Checklist is a shared responsibility between AppSec and Engineering

Ensures...
- Security activities start on-time
- Goals & expectations are aligned
- Clarity on perceived risks
- AppSec process consistency

- Acts as a single source of truth for information about the activity’s details
Things Are Hacked...
The Security Defect Workflow: Verification

1. Vulnerability Reported

   - Behavior Replicated?
     - Yes → Root Cause Analysis Task Creation
     - No → More Information Requested

   - Await Additional Information
The Security Defect Workflow: Resolution
One Report; Many Benefits

- **Perspective:** A formal deliverable sets the tone for a level of quality & completeness of the work
- **Context:** Holistic view of key activity properties
- **Compliance:** Report aggregates necessary information needed for auditors and customers
- **Historic Value:** Easily allows differential analysis of year-over-year results for a given codebase
- **Debrief:** Ensures that all stakeholders have the complete picture of the security activity’s output
In Conclusion
Building a Good Program is Hard to Do...

- **Growing in maturity is difficult when you’re firefighting**
  - Have a strategic role to focus on the bigger picture

- **Buy-in for initiatives is tough without great relationships**
  - Trust & rapport with key stakeholders is necessary for wins

- **It can be easy to lose sight of “the mission at hand”**
  - AppSec exists to make engineering more successful

- **There’s always so much to do — and it’s all important!**
  - Prioritize early & often, track coverage, and be realistic
Highlights of Our 2018 Planning

- Big increase of automated continuous security testing
- A new software inventory with focus on security metadata
- In-house creation of more hands-on security training
- Raising our AppSec-to-Engineer team ratio to 10%+
- Broad involvement in the application security community
Thank You!

Matthew Fanto
Email: mfanto@duo.com
Twitter: @mfanto