

Intel Product Security Maturity Model

Intel Security Group, Product Security Group

8 March 2016

Table of Contents

[1 Purpose 4](#_Toc445228722)

[2 Scope 4](#_Toc445228723)

[3 Overview 4](#_Toc445228724)

[3.1 Parameters 5](#_Toc445228725)

[3.2 Maturity Levels 5](#_Toc445228726)

[3.3 Benefits 6](#_Toc445228727)

[4 PSMM Operational Levels 7](#_Toc445228728)

[4.1 Program Sponsorship and Adoption 7](#_Toc445228729)

[4.2 Resources (Staffing & Tools) 9](#_Toc445228730)

[4.3 SDL 11](#_Toc445228731)

[4.4 PSIRT 12](#_Toc445228732)

[4.5 Policy 13](#_Toc445228733)

[4.6 Security Process Development 14](#_Toc445228734)

[4.7 Security Training Courses 15](#_Toc445228735)

[4.8 Reporting and Tracking Tools 17](#_Toc445228736)

[5 PSMM Technical Levels 20](#_Toc445228737)

[5.1 Security Requirements Plan / Security Definition of Done 20](#_Toc445228738)

[5.2 Architecture and Design Reviews 21](#_Toc445228739)

[5.3 Threat Modeling 22](#_Toc445228740)

[5.4 Security Testing 24](#_Toc445228741)

[5.5 Static Analysis Tools 25](#_Toc445228742)

[5.6 Dynamic Analysis (Web Apps) Tools 27](#_Toc445228743)

[5.7 Fuzz Testing Tools 28](#_Toc445228744)

[5.8 Vulnerability Scans / Penetration Testing 30](#_Toc445228745)

[5.9 Manual Code Reviews 31](#_Toc445228746)

[5.10 Secure Coding Standards 32](#_Toc445228747)

[5.11 Open Source / 3rd Party COTS Library Assessment 33](#_Toc445228748)

[5.12 Privacy 34](#_Toc445228749)

[6 Scoring 35](#_Toc445228750)

[7 Validation 36](#_Toc445228751)

[8 Metrics 37](#_Toc445228752)

[8.1 Spider Diagrams 37](#_Toc445228753)

[8.2 Stock Charts 38](#_Toc445228754)

[8.3 Trend Charts 39](#_Toc445228755)

[9 Mappings to Other Maturity Models 40](#_Toc445228756)

[9.1 BSIMM 40](#_Toc445228757)

[9.2 SAMM 44](#_Toc445228758)

[9.3 DFS 48](#_Toc445228759)

[9.4 Microsoft SDL Optimized Model 53](#_Toc445228760)

[9.5 ISO 27034 54](#_Toc445228761)

[10 Definitions and Abbreviations 57](#_Toc445228762)

[11 Document Attributes 58](#_Toc445228763)

# Purpose

The Intel Product Security Maturity Model (PSMM) is used to measure **how well** each engineering team has adopted security into their overall Security Development Lifecycle (SDL). It was designed and first implemented by the Intel Security Group (ISecG) within Intel Corp. Each of the PSMM parameters support ISecG’s SDL. The Intel Security Product Security Group (PSG) and the engineering teams will maintain scorecards for how mature their products and product teams are for each parameter. Scorecards are updated at least semi-annually and in some product groups, monthly.

# Scope

The goal is for this simple, yet powerful, model to be adopted and used inside Intel company-wide. It can supplement or possibly replace Intel’s DFS Framework. Measuring SDL maturity is one of the top 6 governance metrics on Intel’s product security governance dashboard. Don’t worry about perfect data. You have to start somewhere.

Applies to all Intel Security engineering teams. Managed by the PSG. Data is collected by the Product Security Champions (PSCs). This model is also used by several other Intel business units, such as the Data Center Group (DCG).

Note that in ISecG a Product Security Champion (PSC) is equivalent to three separate roles in the other Intel business units:

1. Intel Product Security Champion [Operational]
2. Intel Product Security Expert [Technical]
3. Intel Security Architect [Authoritative]

# Overview

Intel has a similar program called Design For Security (DFS). Intel has also used Cigital’s BSIMM maturity model. Intel Security was assessed against BSIMM v5 in December 2013. We have since created our own simpler product security maturity model which better mirrors our SDL. The PSMM was developed by Intel Security (McAfee) security engineers who live and practice these activities every day. It is grass roots.

## Parameters

The Intel Security PSMM has 20 parameters. Each parameter is categorized as either operational or technical. Operation parameters cover the maturity of the SDL program and how it operates. Technical parameters cover the maturity of software architects and engineers and the tools used.

**Operational**

1. Program
2. Resources
3. SDL
4. PSIRT
5. Policy
6. Process
7. Training
8. Reporting / Tracking Tools

**Technical**

1. Security Requirements Plan / Security Definition of Done
2. Architecture and Design Reviews
3. Threat Modeling
4. Security Testing
5. Static Analysis
6. Dynamic Analysis
7. Fuzz Testing
8. Vulnerability Scans / Penetration Testing
9. Manual Code Reviews
10. Secure Coding Standards
11. Open Source / 3rd Party COTS Libraries
12. Privacy

## Maturity Levels

The PSMM defines five (5) different maturity levels. They are, from least mature to most mature:

1. None [Initial]
2. Initial [Repeatable]
3. Basic [Defined]
4. Acceptable [Managed]
5. Mature [Optimized]

The desired level for the organization, all engineering teams and products is level 5-Mature. Be aware that the effort it takes to move from 4-Acceptable to 5-Mature may be several times that of getting from 1-None to 4-Acceptable, so get there gradually. The levels in brackets [ ] are another way of stating the same levels. Each PSMM parameter has well defined conditions for determining each level.

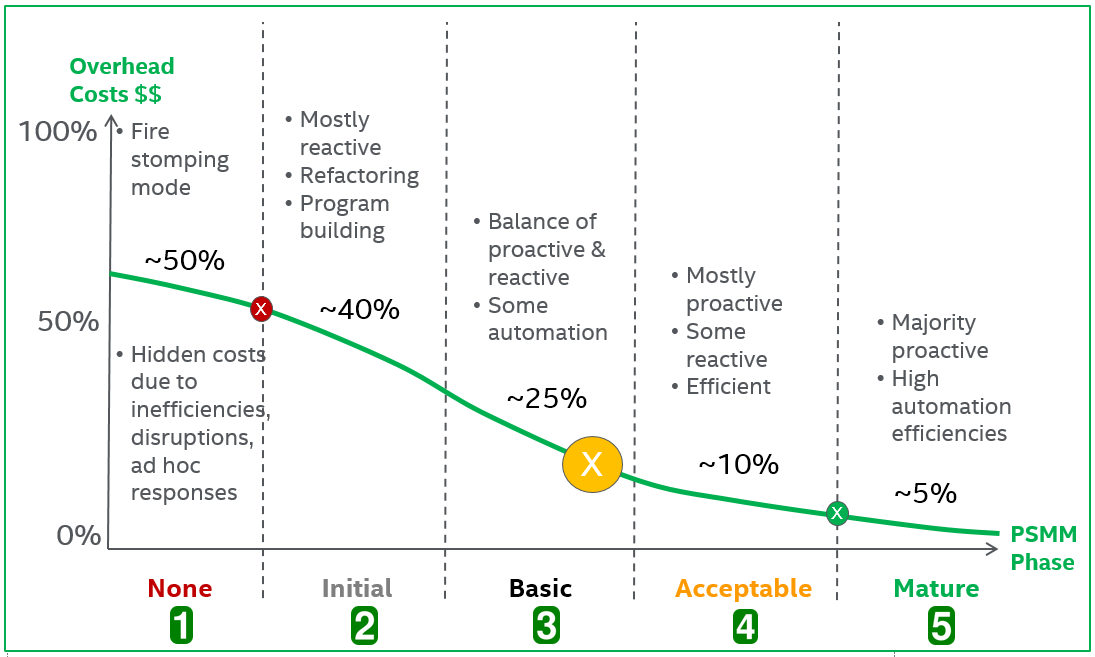
Terminology from [Bruce Tuckman](http://en.wikipedia.org/wiki/Tuckman%27s_stages_of_group_development)’s “Forming-Storming-Norming-Performing” stages of development model is used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Maturity Level** | **Tuckman's Stages of Development** | | **PSMM Score** |
| 1 | **None** | **None** | Ad-hoc. | 0% - 25% |
| 2 | **Initial** | **Forming** | Team initiation phase. | 25% - 50% |
| 3 | **Basic** | **Storming** | Various ideas compete, often fiercely, for consideration. | 50% - 75% |
| 4 | **Acceptable** | **Norming** | Rules, values, behavior, methods, tools are being established. | 75% - 85% |
| 5 | **Mature** | **Performing** | The interpersonal structure becomes the tool of task activities. Roles become flexible and functional. Group energy is channeled into the task. | 85% - 100% |

**Table 3.2.1 Intel PSMM - Stages of Development**

## Benefits

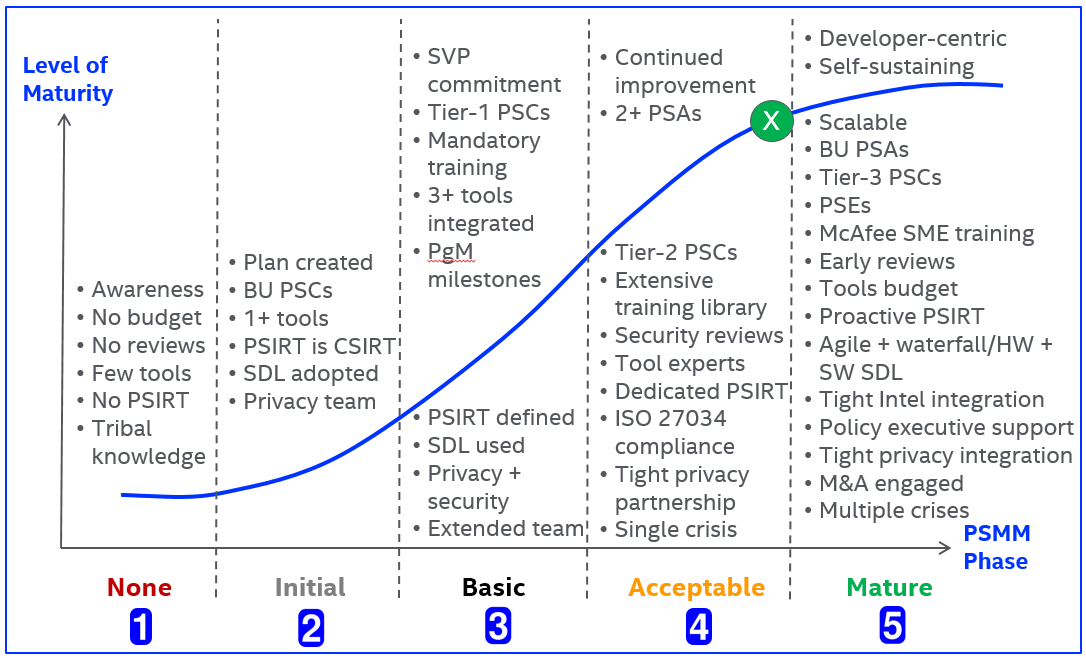
The benefits of having a mature program are lower overall cost and higher performing teams. A mature team is highly proactive and automated. An immature team is inefficient, ad hoc, and stressed.



**Figure 3.3.1 Intel PSMM - Approximate Overhead Percentage Cost per Phase**

# PSMM Operational Levels

There are eight (8) operational parameters. Most apply to the overall Product Security Group. The others apply to each engineering product group. Each product group will be at different maturity levels for each parameter. The combined maturity scores place them and the organization into an overall maturity category.



**Figure 4.1 Intel PSMM - Operational**

## Program Sponsorship and Adoption

This parameter measures how well the product security program is supported by executive management, how well it is staffed and organized.

**Level 1 None**

* No Product Security Group (PSG) program exists
* The following programs either do not exist or are implemented with less than 25% adoption:
  + Product Security Incident Response Team (PSIRT)
  + Product Security Lifecycle Framework (Agile SDL/S-PLF) methodologies
  + Privacy Impact Assessment (PIA)
  + Security ISO Certification or Accreditation
* Extended team consists of who you know

**Level 2 Initial**

* Aware and committed to adoption across the BU
* The following programs are forming with at least 25% adoption:
  + Product Security Group (PSG)
  + Product Security Incident Response Team (PSIRT)
  + Product Security Lifecycle Framework (Agile SDL/S-PLF) methodologies
  + Privacy Impact Assessment (PIA)
  + Security ISO Certification or Accreditation
* Extended Team: Internal stakeholders and partners identified
* BSIMM-SM1.3: Educate executives

**Level 3 Basic**

* Product VPs committed to adoption across all products within BUs
* The following programs are storming with at least 50% adoption:
  + Product Security Group (PSG)
  + Product Security Incident Response Team (PSIRT)
  + Product Security Lifecycle Framework (Agile SDL/S-PLF) methodologies
  + Privacy Impact Assessment (PIA)
  + Security ISO Certification or Accreditation
* Extended Team holds weekly product vulnerability status meetings
* BSIMM-SM1.6: Require security sign-off

**Level 4 Acceptable**

* Demonstrates BUs’ continued improvement efforts, community contribution, and leadership in product security
* The following programs are norming within 75% or better adoption:
  + Product Security Group (PSG)
  + Product Security Incident Response Team (PSIRT)
  + Product Security Lifecycle Framework (Agile SDL/S-PLF) methodologies
  + Privacy Impact Assessment (PIA)
  + Security ISO Certification or Accreditation
* Extended Team can handle a single extended team crisis
* BSIMM-SM2.3: Create or grow a satellite
* BSIMM-CP2.2: Require security sign-off for compliance-related risk
* BSIMM-SR2.2: Create a standards review board

**Level 5 Mature**

* Build and manage a developer-centric, self-sustaining, scalable product security program that is part of quality, includes privacy, using limited resources
* The following programs are performing with 85% to 100% sustainable adoption:
  + Product Security Group (PSG)
  + Product Security Incident Response Team (PSIRT)
  + Product Security Lifecycle Framework (Agile SDL/S-PLF) methodologies
  + Privacy Impact Assessment (PIA)
  + Security ISO Certification or Accreditation
* Extended Team can handle multiple crises
* BSIMM-CP3.3: Drive feedback from SSDL data back to policy

## Resources (Staffing & Tools)

This parameter measures how well the product security program is staffed by PSCs and what tools are available.

**Level 1 None**

* Not properly resourced
* No dedicate headcount budgeted, or no time dedicated to existing resources for security related initiatives
* May (OR) may not have a dedicated PSC resources allotted time for security related initiatives. (Ideally 50% of effort or better allocated.)
* May not have any security tools integrated within their build environments
* No specific security tools funded, or standardized throughout the organization

**Level 2 Initial**

* Have a Product Security Champion (PSC) for each product group (PG)
* Partially dedicated PSC resources allotted time for security related initiatives. (Ideally 50% of effort or better allocated.)
* Have at least one (1) product security tool integrated within their build environments
* Some specific security tools funded and integrated with nightly build automation, or standardized for common re-use across teams

**Level 3 Basic**

* Have a PSC for each Tier-1 product
* Incremental new PSC headcount budgeted, sustainable resources allotted 50% or more time for security related initiatives
* Partially (75% or better) prioritized PSC resources allotted time for security related initiatives
* Required security tools integrated in supported build environment
* Sustainable budget allocated to security tools, or standardized for common re-use across teams
* BSIMM-SM1.2: Create evangelism role and perform internal marketing
* BSIMM-SR1.2: Create a security portal

**Level 4 Acceptable**

* Have a PSC for each Tier-1 & Tier-2 product
* Fully staffed PSC headcount budgeted, dedicated resources allotted 100% time for Tier-1 & Tier-2 security related initiatives
* Fully (100%) prioritized PSC resources allotted time for security related initiatives
* Trained experts on a core set of product security tools
* Defined strategy and roadmap allocated to synchronizing security tool maintenance, standardized tools funded, staffed throughout the organization
* BSIMM-AA2.3: Make SSG available as AA resource or mentor

**Level 5 Mature**

* Have a seasoned PSG Product Security Architect (PSA) dedicated to each R&D PG
* Have a dedicated PSC for all products, including Tier-3
* Have a Product Security Evangelist (PSE) program
* Fully staffed PSC headcount budgeted, dedicated resources allotted 100% time for security related initiatives
* Must have (100%) prioritized PSC resources allotted time for security related activities on a sustainable basis
* Necessary tools budgeted for, centrally supported, sponsored and applied appropriately
* Defined strategy and roadmap allocated to synchronizing security tool maintenance, standardized tools funded, staffed throughout the organization
* BSIMM-AA3.1: Have software architects lead design review efforts

## SDL

This parameter measures how well the SDL is implemented and conforms to ISO 27034.

**Level 1 None**

* Standard SDLC defined and used, security varies by engineer
* Security is reactive
* Customer risk is undefined

**Level 2 Initial**

* Adopt an SDL
* Security is passive
* Customer risk is understood

**Level 3 Basic**

* SDL defined, published and used, engineering trained
* Security is proactive
* Customer risk is monitored
* BSIMM-SM1.4: The software security process will involve release gates/checkpoints

**Level 4 Acceptable**

* Complies with ISO 27034, more proactive activities, less reactive activities
* An Exceptions or “Escapes” process is used for justified, but intentional non-compliance
* Security is integrated
* Customer risk is controlled

**Level 5 Mature**

* Adapted to agile and waterfall, HW/SW/Cloud/IoT, high maturity level scores
* High maturity level scores
* Security is specialized
* Customer risk is minimized

## PSIRT

This parameter measures how well the company’s brand and customers are protected from externally reported product vulnerabilities. PSIRT = Product Security Incident Response Team.

Internal PSIRT SLA’s:

* P1 – Critical = 8.5 -> 10.0 “Must fix immediately (1 day)”
* P2 – High = 7.0 -> 8.4 “Must fix within week (1 week)”
* P3 – Medium = 4.0 -> 6.9 “Must fix within month (1 month)”
* P4 – Low = 0.0 -> 3.9 “Must fix within next major version (1-3 quarters)”

**Level 1 None**

* No incident response procedures or team

**Level 2 Initial**

* Setup and establish a partnership with the Computer Security Incident Response Team (CSIRT)
* Security Architects become PSCs and form an early warning system
* BSIMM-CMVM1.1: Create or interface with incident response

**Level 3 Basic**

* Crisis management procedures defined and used
* PSCs trained on Security Bulletin creation
* Must be able to achieve all PSIRT SLA response times identified
* BSIMM-CMVM1.2: Identify software defects found in operations monitoring and feed them back to development

**Level 4 Acceptable**

* Dedicated PSG-managed team with well-defined procedures
* PSCs create quality Security Bulletins
* Must be able to consistently achieve all PSIRT SLA response times identified
* BSIMM-CMVM2.1: Have emergency codebase response
* BSIMM-CMVM2.2: Track software bugs found in operations through the fix process

**Level 5 Mature**

* 24x7 coverage integrated with entire company
* PSCs are fast, accurate, and follow process
* Consistently achieve all PSIRT SLA response times identified
* BSIMM-CMVM3.1: Fix all occurrences of software bugs found in operations
* BSIMM-CMVM3.2: Enhance the SSDL to prevent software bugs found in operations
* BSIMM-CMVM3.3: Simulate software crisis
* BSIMM-CMVM3.4: Operate a bug bounty program (optional)

## Policy

This parameter measures how well the product security policies are written and adhered to.

**Level 1 None**

* Tribal knowledge

**Level 2 Initial**

* Written policies and procedures
* BSIMM-CP1.3: Create policy

**Level 3 Basic**

* Well documented policies and procedures published
* PSCs know and follow the policies and processes
* BSIMM-SR1.3: Translate compliance constraints to requirements

**Level 4 Acceptable**

* All engineers know and follow the security policies and procedures
* Policies are enforced
* Comply with ISO standards, covers ≥95% of day-to-day situations
* Documented BU policies, procedures, guidelines, and standards
* BSIMM-SR2.5: Create SLA boilerplate

**Level 5 Mature**

* Integrated into big picture from top down with executive buy-in
* Self-enforcing (no paper tiger policies)
* Policies and procedures are reviewed at least annually
* Enable, not barrier

## Security Process Development

This parameter measures how well the product security processes are documented and followed.

**Level 1 None**

* No defined “Definition of Done (DoD)” standards for required security activities involved with shipping software
* Ad Hoc At Best – Inconsistent waterfall processes, or product was a recent M&A product integration
* Peer Reviews & Code Reviews are not conducted, or done without any security patterns in mind

**Level 2 Initial**

* Security testing is done using a waterfall processes at the end of a release, but moving towards Agile/Scrum methodologies defined
* Long term strategic plan focused on resolving the BU business problem in the security assurance space
* The team is increasing their security awareness of security methodologies and best practices are being defined
* Partially tracked security metrics are reported on and key product indicators are being investigated
* Peer Reviews & Code Reviews are occasionally conducted on high risk areas, or with specific security patterns in mind

**Level 3 Basic**

* Waterfall processes reduced below 40%, Agile/Scrum adoption is 60% or better
* Demonstrates BUs’ ability to resolve current security assurance issues
* Sustainable security methodologies and best practices adopted
* Sustainable product readiness meetings, artifacts and key product indicators with standardized metrics archived
* BSIMM-SM1.1: Publish process (roles, responsibilities, plan), evolve as necessary

**Level 4 Acceptable**

* Waterfall processes reduced below 10%, Agile/Scrum adoption is 90% or better
* Demonstrates PGs’ ability to independently resolve current and consider future security assurance issues
* Fully documented product security methodologies and best practices adopted
* Fully documented product readiness meetings, artifacts and key product indicators with standardized metrics archived

**Level 5 Mature**

* Waterfall processes reduced below 1%, Agile/Scrum adoption is 99% or better
* Consistently and repeatedly execute process
* PSCs know and follow the processes
* Fully documented product security methodologies and best practices adopted
* Fully documented product readiness meetings, artifacts and key product indicators with standardized metrics archived

## Security Training Courses

This parameter measures how much high quality security training is created or purchased, delivered and utilized.

**Level 1 None**

* Have little or no product security training
* Company Level
  + No dedicated or centralized security training materials available
  + Course attendance may only be through free online webinars that are not tracked or measured
* Business Unit (BU), Product Group (PG) & Product Team (PT) Level
  + May (OR) May Not have a limited amount of security training topics dedicated, or not have any centralized security training materials available

**Level 2 Initial**

* Have identified and use free product security courses
* Training plan created
* Company Level
  + A limited amount of topics dedicated or centralized security training materials available
  + Limited security training is budgeted and attendance is on a case-by-case basis
  + Some minimum courses are required and tracked against improving overall security awareness

**Level 3 Basic**

* Have a defined set of product security courses
* PSCs have completed assigned courses
* Company Level
  + Sustainable security training is budgeted and attendance is on a controlled team-by-team basis. (e.g. Security Conferences)
  + Sustainable training courses are required and tracked against improving repeatable security patterns
* Business Unit (BU), Product Group (PG) & Product Team (PT) Level
  + Sustainable quarterly MBO’s dedicated on security training materials available
* BSIMM-T1.5: Deliver role-specific advanced curriculum (tools, technology stacks, bug parade)
* BSIMM-T1.6: Create and use material specific to company history
* BSIMM-T1.7: Deliver on-demand individual training

**Level 4 Acceptable**

* Major topics are delivered and internal processes are covered
* PSCs are taking vendor courses
* Company Level
  + Pro-active quarterly ROI metrics of security training materials available
  + Pro-active training courses are targeted preventing repeatable security patterns
  + Sustainable security training is budgeted and attendance is on a controlled team-by-team basis. (e.g. Security Conferences)
  + Sustainable training courses are required and tracked against improving repeatable security patterns
* Business Unit (BU), Product Group (PG) & Product Team (PT) Level
  + Sustainable security training is maintained across 80% or better of all team members
  + Sustainable quarterly MBO’s dedicated on security training materials available
* BSIMM-T2.5: Enhance satellite through training and events
* BSIMM-T2.6: Include security resources in onboarding
* BSIMM-T2.7: Identify satellite through training

**Level 5 Mature**

* Keep program current
* Intel SecuritySubject Matter Expert (SME) provided training
* Company Level
  + Pro-active quarterly ROI metrics of security training materials available
  + Sustainable security training is maintained across 80% or better of all team members
  + Pro-active training courses are targeted preventing repeatable security patterns
* Business Unit (BU), Product Group (PG) & Product Team (PT) Level
  + Sustainable quarterly MBO’s dedicated on security training materials available
  + Sustainable security training is budgeted and attendance is on a controlled team-by-team basis. (e.g. Security Conferences)
  + Sustainable training courses are required and tracked against improving repeatable security patterns
* BSIMM-T3.1: Reward progression through curriculum (certification or HR)
* BSIMM-T3.2: Provide training for vendors or outsourced workers
* BSIMM-T3.3: Host external software security events
* BSIMM-T3.4: Require an annual refresher
* BSIMM-T3.5: Establish SSG office hours

## Reporting and Tracking Tools

This parameter measures how efficiently data is entered and tracked for security metrics. It also measures the quality of security metrics generated.

**Level 1 None**

* Awareness
* No budget
* Issues and reviews tracked via email
* No metrics

**Level 2 Initial**

* All manual
* Very inefficient
* Incomplete
* Immeasurable
* Limited coverage
* COTS tools used to manage incidents (e.g. Excel spreadsheet)
* Metrics plan created, data is gathered

**Level 3 Basic**

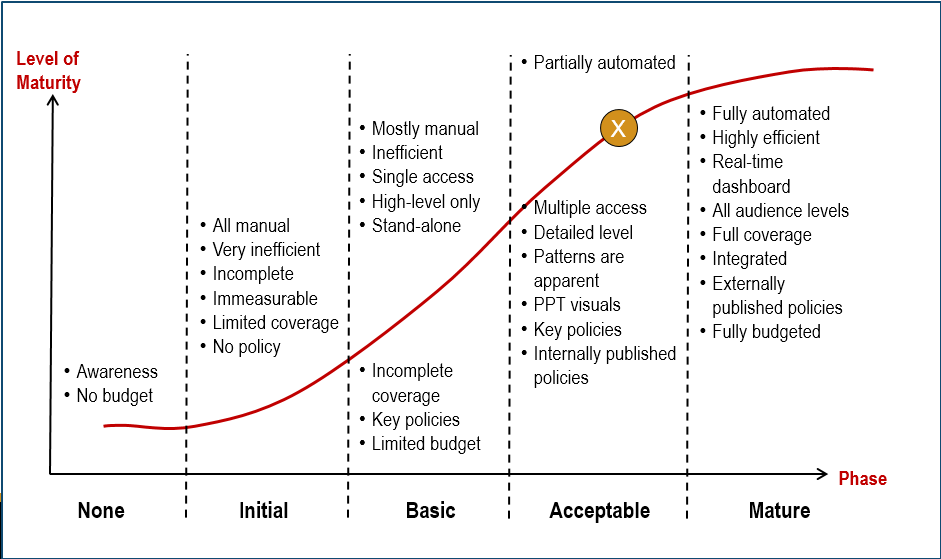
* Mostly manual
* Inefficient
* Single access
* High-level data only
* Stand-alone
* Incomplete coverage
* Limited budget
* Issues and reviews tracked in detailed spreadsheets
* PSCs reporting PSIRT and Security review data
* Metrics include PSIRT, SDL Activities, and Training
* Metrics reported by year
* Metrics granular to the Business Unit (BU) level
* BSIMM-CR1.6: Use centralized reporting to close the knowledge loop and drive training

**Level 4 Acceptable**

* Partially automated
* Multiple access
* Detailed level data
* Patterns are apparent
* PPT visuals
* Internally published policies
* Application or database with configurable dashboard and notification capabilities
* PSCs complete survey for each review
* Metrics include Exceptions, Static Analysis, and Dynamic Analysis
* Metrics by quarter
* Metrics granular to the Product Group (PG) level
* BSIMM-SM2.5: Identify metrics and use them to drive budgets
* BSIMM-CP2.3: Implement and track controls for compliance
* BSIMM-CMVM2.2: Track software bugs found in operations through the fix process
* BSIMM-CMVM2.3: Develop an operations inventory of applications

**Level 5 Mature**

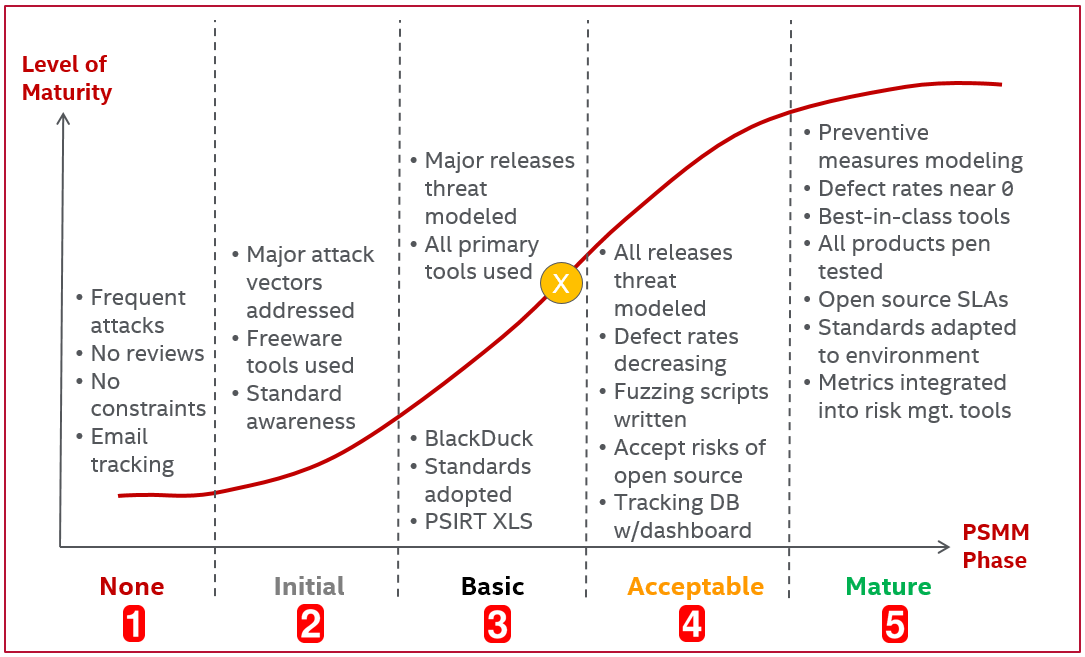
* Fully automated
* Highly efficient
* Real-time dashboard
* All audience levels
* Full coverage
* Integrated
* Fully budgeted
* Data and metrics tightly integrated with IT and executive-level risk management tools, concurrent usage, workflow
* Metrics include all PSMM parameters
* Metrics include trend charts
* Metrics granular to the Product level (heat maps)
* Trends in the metrics can be identified and acted upon
* BSIMM-SM3.1: Use an internal tracking application with portfolio view



**Figure 4.8.1 – Harold’s Product Security Metrics Maturity Model**

# PSMM Technical Levels

There are twelve (12) technical parameters. Each engineering team will be at different maturity levels for each parameter. The combined maturity scores place them into an overall maturity category.



**Figure 5.1.5 Intel PSMM - Technical**

The following parameters are required in Intel’s Agile SDL. Dismissing them with “Not Applicable” automatically generates a process exception which must be recorded and approved by executive management.

* T1 Security Requirements Plan / Definition of Done (DoD)
* T3 Threat Modeling
* T5 Static Analysis
* T12 Privacy

## Security Requirements Plan / Security Definition of Done

This parameter measures how well software security requirements are being performed and verified by both engineering and QA. In the Waterfall SDL this is called the Security Requirements Plan document. In the Agile SDL this is called the Security Definition of Done (DoD) document.

**Level 0 NA**

* Since this is a mandatory SDL activity, selecting the Not Applicable option automatically requires that an Exception be filed

**Level 1 None**

* Ad hoc - local architectural reviews only
* No security plans developed at project start

**Level 2 Initial**

* Standard security requirement tasks defined

**Level 3 Basic**

* PgMs answer “The 7 Key Questions” in the Agile PLF Checklist spreadsheet or other similar risk-based tripwire questions
* BSIMM-SFD1.1: Build and publish security features
* BSIMM-SFD1.2: Engage SSG with architecture

**Level 4 Acceptable**

* Product teams conduct and report on required security tasks as defined in their security plan for their project milestones
* BSIMM-SFD2.1: Build secure-by-design middleware frameworks and common libraries
* BSIMM-SFD2.2: Create SSG capability to solve difficult design problems

**Level 5 Mature**

* Product teams engage their PSCs early
* BSIMM-SFD3.1: Form a review board or central committee to approve and maintain secure design patterns
* BSIMM-SFD3.3: Find and publish mature design patterns from the organization

## Architecture and Design Reviews

This parameter measures how well the software architecture and design are being reviewed early on by engineering’s security architects.

**Level 1 None**

* Architecture designed for performance and maintainability, not security

**Level 2 Initial**

* Informal architectural review conducted by engineering
* Reviews reveal 1+ medium severity issues
* BSIMM-AA1.1: Perform security feature review
* BSIMM-AA1.2: Perform design review for high-risk applications

**Level 3 Basic**

* Architecture and design reviews documented
* Formal architectural review conducted by product/security architects before all major releases
* Reviews reveal 1+ high severity issues
* BSIMM-AA1.3: Have SSG lead design review efforts
* BSIMM-AA1.4: Use a risk questionnaire to rank applications

**Level 4 Acceptable**

* Architecture and design reviews published with restricted access
* Trained security architects oversee frequent reviews accounting for all known attack vectors
* Reviews reveal 2+ high severity issues or the review time spent is deemed worth while
* BSIMM-AA2.1: Define and use AA process
* BSIMM-AA2.2: Standardize architectural descriptions (including data flow)

**Level 5 Mature**

* Separation of privileges and type enforcement address unknown attack vectors
* Reviews reveal multiple high and medium severity issues and the issues are effectively addressed early in the development cycle
* Architecture documents extensive enough to be used for Common Criteria (EAL-3) certification
* BSIMM-AA3.2: Drive analysis results into standard architecture patterns

## Threat Modeling

This parameter measures how well the threat model is developed, used, and maintained.

**Level 0 NA**

* Since this is a mandatory SDL activity, selecting the Not Applicable option automatically requires that an Exception be filed

**Level 1 None**

* No threat model
* Lack of modeling exposed by large number of customer reported vulnerabilities and attacks

**Level 2 Initial**

* Major attack vectors identified and addressed
* Threat models developed
* BSIMM-AM1.1: Build and maintain a top N possible attacks list

**Level 3 Basic**

* Major and typical attack vectors identified and addressed
* Formal threat modeling conducted by product/security architects before all major releases
* Application-specific threat models developed, used, and maintained [SAMM-TA1A]
* Develop attacker profile from software architecture [SAMM-TA1B]
* BSIMM-AM1.2: Create a data classification scheme and inventory
* BSIMM-AM1.3: Identify potential attackers
* BSIMM-AM1.4: Collect and publish attack stories
* BSIMM-AM1.5: Gather attack intelligence
* BSIMM-AM1.6: Build an internal forum to discuss attacks

**Level 4 Acceptable**

* Threat models used and updated frequently
* Major, typical, and minor attack vectors identified and addressed
* Trained security architects oversee frequent threat model reviews accounting for all known attack vectors
* Adopt a weighting system for measurement of threats [SAMM-TA2B]
* Build and maintain abuse-case models per project [SAM-TA2A]
* BSIMM-AM2.1: Build attack patterns and abuse cases tied to potential attackers
* BSIMM-AM2.2: Create technology-specific attack patterns

**Level 5 Mature**

* Threat models updated by advanced team of researchers
* Least privileges and type enforcement address unknown attack vectors
* Explicitly evaluate risk from third-party components [SAM-TA3A]
* Elaborate threat models with compensating controls [SAM-TA3B]
* BSIMM-AM3.1: Have a science team that develops new attack methods
* BSIMM-AM3.2: Create and use automation to do what attackers will do

## Security Testing

This parameter measures how well software security requirements are being performed and verified by both engineering and QA.

**Level 1 None**

* No security plan
* No security plan testing or validation performed

**Level 2 Initial**

* Security plan created
* Security plan testing and validation performed occasionally

**Level 3 Basic**

* Security plan testing and validation performed completely at least once before release
* Functional Testing (what you know) performed to verify intended behavior
* BSIMM-ST1.1: Ensure QA supports edge/boundary value condition testing
* BSIMM-ST1.3: Drive tests with security requirements and security features

**Level 4 Acceptable**

* Security plan testing and validation performed completely several times before release
* Negative Space Testing (what hackers know) performed to identify non-intended behavior
* Threat model updated with issues discovered in testing
* BSIMM-ST2.1: Integrate black box security tools into the QA process
* BSIMM-ST2.4: Share security results with QA

**Level 5 Mature**

* Security plan testing and validation performed continuously and completely both before and after release
* At least 90% of all code paths tested
* BSIMM-ST3.1: Include security tests in QA automation
* BSIMM-ST3.3: Drive tests with risk analysis results
* BSIMM-ST3.4: Leverage coverage analysis
* BSIMM-ST3.5: Begin to build and apply adversarial security tests (abuse cases)

## Static Analysis Tools

This parameter measures how well tools can find vulnerabilities in the source code. Also known as Static Application Security Testing (SAST).

**Level 0 NA**

* Since this is a mandatory SDL activity, selecting the Not Applicable option automatically requires that an Exception be filed

**Level 1 None**

* Use no static analysis tools or use compiler flags only
* Unable to triage and maintain new & unconfirmed issues
* Unable to perform an analysis scan against production code in the build environment that is packaged for customers
* Unable to tune and optimize the tool to reduce false positives
* Unable to report static analysis metrics

**Level 2 Initial**

* Use one or more static analysis tools
* Incorporate a static analysis tool that supports some of the programming languages used within your product
* Triage and maintain all new & unconfirmed issues on a quarterly basis
* Perform an analysis scan against some production code (50%) in the build environment that is packaged for customers
* Tune and optimize the tool on a periodic basis to reduce false positives
* Run the static analysis tools manually
* Reporting static analysis metrics

**Level 3 Basic**

* Static analysis runs automatically with builds
* Must incorporate a static analysis tool that supports all of the major programming languages used within your product
* Must triage and maintain all new & unconfirmed issues on a monthly basis
* Must perform an analysis scan against most production code (80%) in the build environment that is packaged for customers
* Must tune and optimize the tool to reduce false positives discovered to prevent unnecessary technical debt
* Run static analysis tools either manually or semi-automatically against the latest releasable code
* Must report multiple static analysis metrics
* BSIMM-CR1.4: Use automated tools along with manual review

**Level 4 Acceptable**

* Majority of products analyzed frequently, defect rate decreasing
* Must incorporate static analysis tools that supports all of the programming languages used within your product
* Must triage and maintain all new & unconfirmed issues on a weekly basis
* Must perform an analysis scan against most all production code (95%) in the build environment that is packaged for customers
* Must tune and optimize the tool on a periodic basis to reduce false positives discovered to prevent unnecessary technical debt
* Engineering team has a resident expert on the static analysis tools used
* Must centrally report static analysis metrics into a rolled up dashboard for cross BU reporting
* BSIMM-CR2.5: Assign tool mentors
* BSIMM-CR2.6: Use automated tools with tailored rules

**Level 5 Mature**

* Defects fixed quickly, real defect rate near zero (0) at Sprint/PSI boundaries
* Must incorporate a single static analysis tool that supports all of the programming languages used within your product
* Must triage and maintain all new & unconfirmed issues on a monthly basis
* Must perform an analysis scan against all production code (99+%) in the build environment that is packaged for customers
* Must tune and optimize the tool on a regular basis to reduce false positives discovered to prevent unnecessary technical debt
* Both engineering and QA teams have at least one resident expert on the static analysis tools used
* Must centrally report static analysis metrics, with normalized data, into a rolled up dashboard for cross BU reporting
* BSIMM-CR3.2: Build a factory
* BSIMM-CR3.4: Automate malicious code detection

## Dynamic Analysis (Web Apps) Tools

This parameter measures how well tools can find vulnerabilities in the binary executables and Web applications. Also known as Dynamic Application Security Testing (DAST).

**Level 1 None**

* User feedback only from their tools
* Unable to incorporate a dynamic analysis tool
* Unable to triage and maintain new & unconfirmed issues
* Unable to perform an analysis scan against production code in the build environment that is packaged for customers
* Unable to tune and optimize the tool to reduce false positives
* Unable to report dynamic analysis metrics

**Level 2 Initial**

* Open source tools used
* Incorporate a dynamic analysis tool that supports some of the programming languages used within the product
* Triage and maintain all new & unconfirmed issues on an semi-annual basis
* Perform an analysis scan against all production code (50%) in the build environment that is packaged for customers
* Tune and optimize the tool on a periodic basis to reduce false positives
* Run the dynamic analysis tools manually
* Report key dynamic analysis metrics

**Level 3 Basic**

* Tool purchased, maintained and used
* Incorporate a dynamic analysis tool that supports all of the major programming languages used within the product
* Triage and maintain all new & unconfirmed issues on a quarterly basis
* Perform an analysis scan against all production code (80%) in the build environment that is packaged for customers
* Tune and optimize the tool to reduce false positives discovered to prevent unnecessary technical debt
* Engineering and QA are trained on dynamic analysis tools used
* Run dynamic analysis tools either manually or semi-automatically against the latest releasable code
* Report multiple dynamic analysis metrics

**Level 4 Acceptable**

* Applicable products analyzed frequently, high and medium severity issues fixed
* Defect rate near zero (0) in finished product
* Incorporate a dynamic analysis tool that supports most all of the programming languages used within the product
* Triage and maintain all new & unconfirmed issues on a monthly basis
* Perform an analysis scan against most all production code (95%) in the build environment that is packaged for customers
* Tune and optimize the tool on a periodic basis to reduce false positives discovered to prevent unnecessary technical debt
* Engineering team has a resident expert on the dynamic analysis tools used
* Incorporate the dynamic analysis tools into a continuous integration environment for nightly build validation against the latest releasable code
* Centrally report all relevant dynamic analysis metrics into a rolled up dashboard for cross product group reporting
* BSIMM-ST2.1: Integrate black box security tools into the QA process
* BSIMM-CR2.5: Assign tool mentors

**Level 5 Mature**

* Defects fixed quickly, defect rate near zero (0) at Sprint/PSI boundaries
* Incorporate a dynamic analysis tool that supports all of the programming languages used within the product
* Triage and maintain all new & unconfirmed issues on a weekly basis
* Perform an analysis scan against all production code (99+%) in the build environment that is packaged for customers
* Tune and optimize the tool on a regular basis to reduce false positives discovered to prevent unnecessary technical debt
* Both engineering and QA teams have at least one resident expert on the dynamic analysis tools used
* Incorporate the dynamic analysis tools into a continuous integration environment for nightly build validation against all of the product’s code, and acting on the found defects
* Centrally report dynamic analysis metrics, with normalized data, into a rolled up dashboard for cross product group reporting
* BSIMM-CR3.2: Build a factory

## Fuzz Testing Tools

This parameter measures how well tools can find vulnerabilities and system crashes in the running binary executables by probing all inputs, network protocols, and file formats.

**Level 1 None**

* Crashes discovered randomly
* No fuzz testing tools used
* Unable to triage and maintain all new & unconfirmed issues
* Unable to perform a fuzz test against production executables
* Unable to report fuzz testing metrics

**Level 2 Initial**

* Free/open source tools used by engineers / SDET (e.g. Peach Fuzzer, Codenomicon)
* Incorporate fuzz testing tools that support some of the attack vectors/inputs
* Triage and maintain all new & unconfirmed issues on a quarterly basis
* Perform a fuzzing scan against most production executables in a test environment
* Little or no tuning of the tool to reduce false positives
* Run the fuzz testing tools manually
* Report key fuzz testing metrics
* ISO-27034: A.9.5 Verification – 12. Fuzz Testing

**Level 3 Basic**

* Tool purchased, maintained and used
* Incorporate fuzz testing tools that support the highest risk attack vectors/inputs
* Triage and maintain all new & unconfirmed issues on a monthly basis
* Perform fuzz testing against most production executables (80%) in the test or build environments
* Provide some intelligent tuning of the tool to reduce false positives
* Run fuzz testing tools either manually or semi-automatically
* Report multiple fuzz testing metrics
* Engineering and QA are trained on the fuzz testing tools used

**Level 4 Acceptable**

* Applicable products fuzz tested frequently, high and medium severity issues fixed
* Defect rate near zero (0) in finished product
* Incorporate fuzz testing tools that support all known attack vectors/inputs
* Triage and maintain all new & unconfirmed issues on a weekly basis
* Perform fuzz test against most all production executables (95%) in the build environment
* Tune and optimize the tool on a periodic basis to reduce false positives discovered to prevent unnecessary technical debt
* Run fuzz testing tools automatically against the latest releasable code/executables
* Centrally report fuzz testing metrics into a rolled up dashboard for cross product group reporting
* Engineering and QA teams have access to at least one company-wide expert on each fuzz testing tool used
* BSIMM-ST2.1: Integrate black box security tools into the QA process
* BSIMM-CR2.5: Assign tool mentors

**Level 5 Mature**

* Defects fixed quickly, defect rate near zero (0) at Sprint/PSI boundaries
* Incorporate fuzz testing tools that support all known and unknown attack vectors/inputs
* Triage and maintain all new & unconfirmed issues continuously
* Perform fuzz testing against all production executables (99+%) in the build environment that is packaged for customers
* Tune and optimize the tool on a regular basis to reduce false positives discovered to prevent unnecessary technical debt
* Custom fuzz tests are written specifically for the product’s environment
* Incorporate the fuzz testing tools into a continuous integration environment for nightly build validation against all of the product’s code/executables, and acting on the found defects
* Centrally report fuzz testing metrics, with normalized data, into a rolled up dashboard for cross product group reporting
* Engineering and QA teams have access to at least one resident expert on each fuzz testing tool used
* BSIMM-ST3.2: Perform fuzz testing customized to application APIs

## Vulnerability Scans / Penetration Testing

This parameter measures how well tools and humans can find vulnerabilities in the binary executables and the environment they are deployed on.

**Level 1 None**

* Vulnerability discovery escalated by customers

**Level 2 Initial**

* Tools identified and used by engineering / SDET
* Free tools acquired
* BSIMM-PT1.1: Use external penetration testers to find problems

**Level 3 Basic**

* Vulnerability scans occasionally performed, defects analyzed
* Enterprise class tools budgeted for
* Defects tracked in Bugzilla
* BSIMM-PT1.2: Feed results to the defect management and mitigation system
* BSIMM-PT1.3: Use penetration testing tools internally

**Level 4 Acceptable**

* Vulnerability scans regularly performed, defects analyzed
* Resident pen testing expert available
* Enterprise class tools purchased, deployed and used
* BSIMM-PT2.2: Provide penetration testers with all available information
* BSIMM-PT2.3: Schedule periodic penetration tests for application coverage
* BSIMM-CR2.5: Assign tool mentors

**Level 5 Mature**

* Multiple vulnerability scanning tools automatically run regular scans, defects fixed
* Every release pen tested, defects fixed before launch
* BSIMM-PT3.1: Use external penetration testers to perform deep-dive analysis
* BSIMM-PT3.2: Have the SSG customize penetration testing tools and scripts

## Manual Code Reviews

This parameter measures how well humans can find vulnerabilities in the source code.

**Level 1 None**

* No manual code reviews conducted or ad-hoc

**Level 2 Initial**

* Aware of standards, occasional adherence
* Annual manual code reviews conducted
* BSIMM-CR1.1: Create a top N bugs list (real data preferred)
* BSIMM-CR1.2: Have SSG perform ad hoc review

**Level 3 Basic**

* Manual code reviews conducted on risky new code by multiple engineers
* Periodic manual code reviews for security weaknesses
* BSIMM-CR1.4: Use automated tools along with manual review

**Level 4 Acceptable**

* Manual code reviews conducted on all potentially risky code using a shared tool
* Manual code reviews conducted for security weaknesses before product ships
* BSIMM-CR1.5: Make code review mandatory for all projects

**Level 5 Mature**

* Conducted regularly using a code sharing collaboration tool (e.g. SmartBear Collaborator)
* BSIMM-CR3.2: Build a factory
* BSIMM-CR3.3: Build a capability for eradicating specific bugs from the entire codebase
* BSIMM-CR3.4: Automate malicious code detection

## Secure Coding Standards

This parameter measures how well humans are implementing secure coding standards and best practices.

**Level 1 None**

* No secure coding standards
* Individuals have own coding standards

**Level 2 Initial**

* Aware of standards, occasional adherence
* Individual product teams have own coding standards

**Level 3 Basic**

* Adopted appropriate standards
* Individual product groups have own coding standards
* BSIMM-SR1.1: Create security standards
* BSIMM-SR1.4: Use secure coding standards

**Level 4 Acceptable**

* Following adopted standards
* Intel BUs have own coding standards
* BSIMM-SR2.3: Create standards for technology stacks
* BSIMM-CR2.2: Enforce coding standards
* BSIMM-SE2.4: Use code signing

**Level 5 Mature**

* Standards are integrated into manual code reviews and static analysis
* Intel has common coding standards
* Contributing to standards
* BSIMM-SFD3.2: Require use of approved security features and frameworks
* BSIMM-SE3.2: Use code protection

## Open Source / 3rd Party COTS Library Assessment

This parameter measures how well we secure code and commercial off-the-shelf (COTS) libraries not developed by us.

**Level 1 None**

* Selected by any engineer, used with no constraints
* No Product Security 3rd Party Library Assessments are being maintained pro-actively, most likely being updated once there is a customer escalation driving a security issue to be fixed

**Level 2 Initial**

* Manually maintain lists of used
* None being maintained pro-actively, most likely being updated once there is a customer escalation driving a security issue to be fixed
* Moving toward documenting all 3rd party libraries and versions shipped across all supported releases
* BSIMM-CP2.4: Paper all vendor contracts with software security SLAs

**Level 3 Basic**

* Run inventory tools (e.g. Black Duck)
* Now fully maintaining all documented 3rd party libraries and versions shipped across all supported releases
* Now planning all 3rd party library "Min-Ship" changes in the initial Planning or Concept Commit Phase, and before the next release starts development
* Documentation covers when any libraries may be due to reach EOL status, before the supported product is EOL
* Pro-active scanning and monitoring of the versions against known Common Vulnerabilities and Exposures (CVE)
* BSIMM-SR2.4: Identify open source

**Level 4 Acceptable**

* Know and accept the inherited security risks
* Now fully maintaining all documented 3rd party libraries and versions shipped across all supported releases
* Now planning all 3rd party library "Min-Ship" changes in the initial Planning or Concept Commit Phase, and before the next release starts development
* Documentation covers when any libraries may be due to reach EOL status, before the supported product is EOL
* Pro-active scanning and monitoring of the versions against known Common Vulnerabilities and Exposures (CVE)
* BSIMM-SR3.1: Control open source risk

**Level 5 Mature**

* Partner SLAs and maintenance contracts
* Now fully maintaining all documented 3rd party libraries and versions shipped across all supported releases
* Now planning all 3rd party library "Min-Ship" changes in the initial Planning or Concept Commit Phase, and before the next release starts development
* Documentation covers when any libraries may be due to reach EOL status, before the supported product is EOL
* Pro-active scanning and monitoring of the versions against known Common Vulnerabilities and Exposures (CVE)
* BSIMM-CP3.2: Impose policy on vendors
* BSIMM-SR3.2: Communicate standards to vendors

## Privacy

This parameter measures how well we are protecting the confidentiality or privacy of personal and corporate data.

**Level 0 NA**

* Since this is a mandatory SDL activity, selecting the Not Applicable option automatically requires that an Exception be filed

**Level 1 None**

* Privacy is not included with product security

**Level 2 Initial**

* Privacy team engaged
* All new products conduct a privacy review

**Level 3 Basic**

* Privacy works alongside product security
* Personally Identifiable Information (PII) defined and identified
* Privacy Impact Assessments (PIA) defined
* Maintain well established baselines for privacy
* BSIMM-CP1.2: Identify PII obligations

**Level 4 Acceptable**

* Privacy is integrated with product security
* All products conduct a privacy review
* Privacy Impact Assessments (PIA) are conducted when PII is involved
* All relevant privacy regulations are addressed in the product
* BSIMM-CP2.1: Identify PII data inventory
* BSIMM-CP2.5: Promote executive awareness of compliance and privacy obligations

**Level 5 Mature**

* Threat Model for data and PII is performed
* Product security implies privacy
* QA implies quality, security and privacy
* BSIMM-SR2.5: Create SLA boilerplate

# Scoring

PSCs should score their products starting with a product tab on the PSMM spreadsheet. They simply need to answer 10-20 questions from the drop-down lists. If a more detailed description of each level of each parameter is needed, then refer to this PSMM Word doc.

Not every requirement listed in each level for each parameter must be done to be considered as acting at that level. If most of the items are followed then consider your product to be at that level. This mainly applies to parameter levels with 3+ activities listed.

Scores should be calculated from the following different data levels. Examples are in parenthesis:

1. Company Level (Intel Corp.)
2. Company BU Level (ISecG)
3. Product Group Level (Security Management)
4. Product Level (ePO)
5. Agile Team Level (4-9 individuals)
6. Parameter Level (e.g. Static Analysis)

With 20 PSMM parameters and a score ranging from 1-5 for each, we simply add up all of the scores to determine the overall PSMM score. Doing so gives us a minimum total score of 20% and a maximum total score of 100%.

For determining consolidated scores for product group and company levels, simple averages will be used. All scores are weighted equally.

PSMM scores for just the Technical parameters may also be calculated without regard for the Operational parameters, and vice versa. When doing so, scores are weighted equally so they add up to 100%.

Use the PSMM spreadsheet with drop-downs for each of the 20 parameters to automatically calculate your current PSMM score.

|  |  |  |  |
| --- | --- | --- | --- |
| **PSMM Level** | **Min. Score** | **Max. Score** | **Considered “In” Score** |
| 1-None | 20 | 39 | 20-29 |
| 2-Basic | 40 | 59 | 30-49 |
| 3-Initial | 60 | 79 | 50-69 |
| 4-Acceptable | 80 | 99 | 70-89 |
| 5-Mature | 100 | 100 | 90-100 |

**Table 6.1 - PSMM Scoring Table**

# Validation

Individual PSCs should score their own products. If they do not know the answers then they should engage their product teams to get accurate answers. PSC PG Leads should score their Product Group from their perspective.

To maintain objectivity and consistency, PSCs from one Product Group will be assigned to review metrics from a different Product Group.

PGs should review the product and PG scores to identify and correct gross inaccuracies.

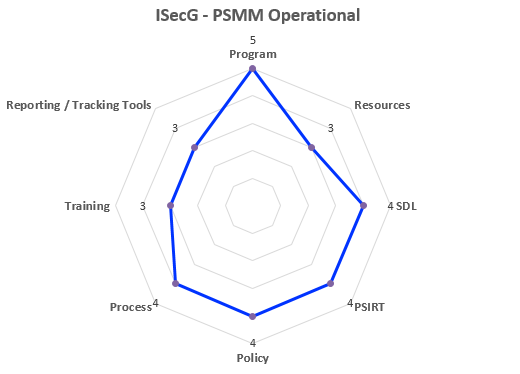
The Product Security and Privacy Governance team will perform rolling audits to ensure compliance and accuracy. Trust but verify.

# Metrics

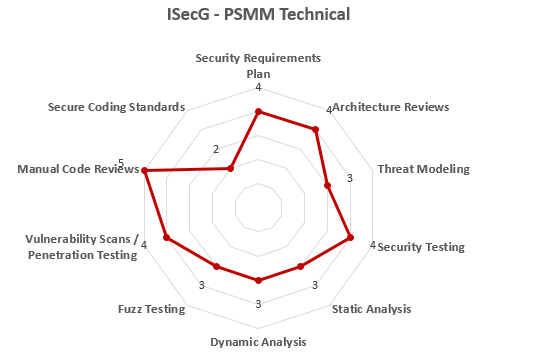
The following chart types are well suited for reporting product security maturity model metrics.

## Spider Diagrams

Spider or radar diagrams are ideal for showing all operational and technical parameters for a single entity.



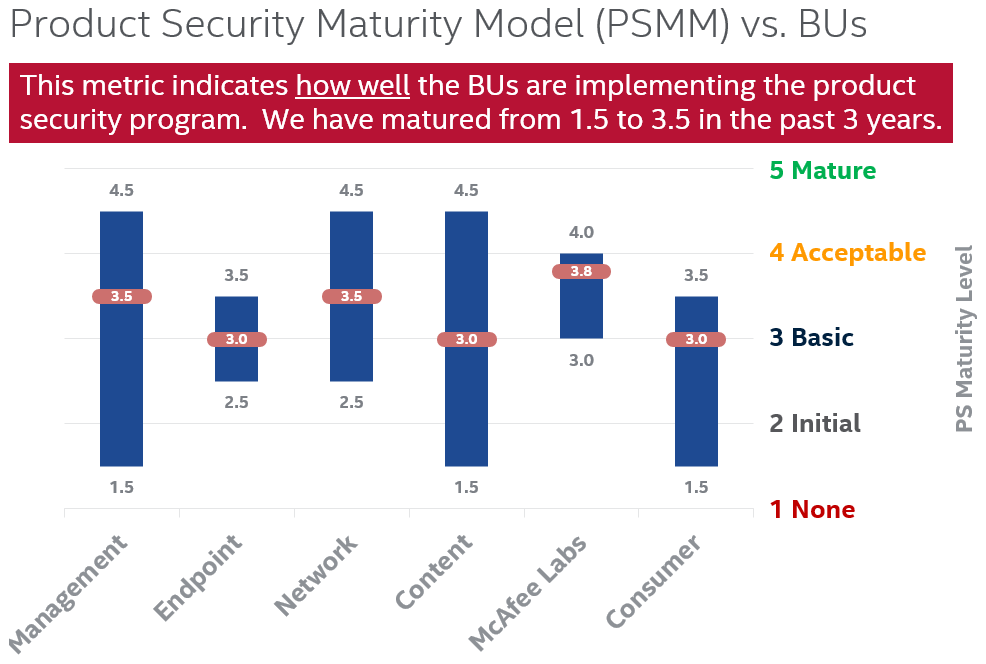
**Figure 8.1.1 – Example Spider Diagram – Operational**



**Figure 8.1.2 – Example Spider Diagram - Technical**

## Stock Charts

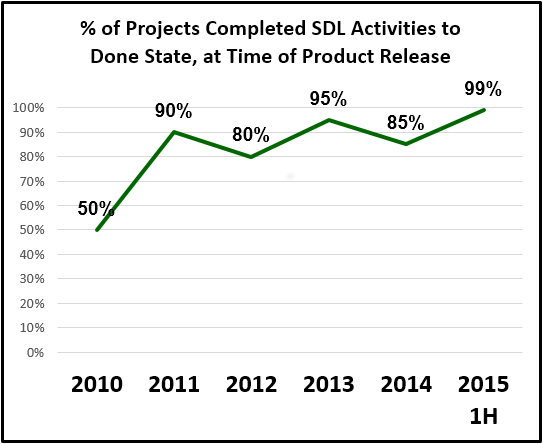
Stock charts are ideal for comparing teams, product groups and BUs side-by-side since they show the average score along with the worst and best scores.



**Figure 8.2.1 – Example Stock Chart**

## Trend Charts

Trend charts are ideal for showing progress over time.



**Figure 8.3.1 – Example Trend Chart**

# Mappings to Other Maturity Models

Common SDL Maturity Models:

* **BSIMM**: Build Security In Maturity Model - Cigital
* **SAMM**: Software Assurance Maturity Model - OWASP
* **DFS**: Design For Security - Intel
* **PSMM**: Product Security Maturity Model - Intel Security

Common application security standards:

* **ISO 27034**: Application Security Part 1 – ISO/IEC

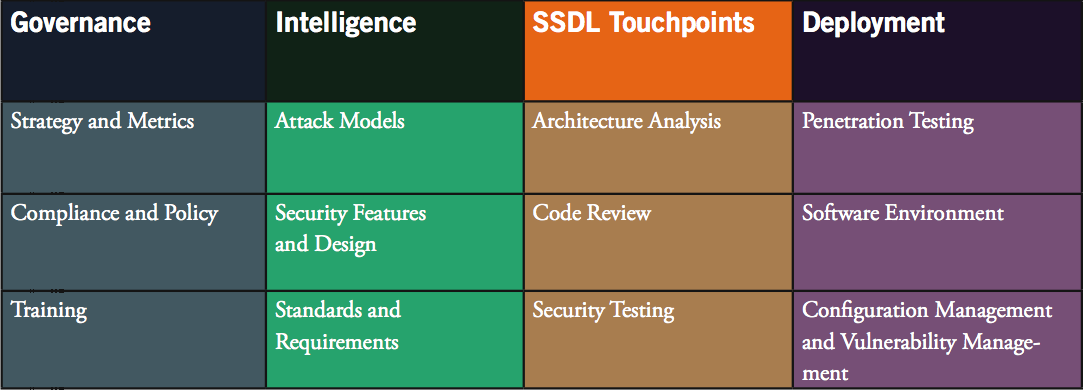
## BSIMM

**BSIMM**: Build Security In Maturity Model

<https://www.bsimm.com>

Developed by Cigital and Fortify. Maintained by Cigital.

Both Intel and Intel Security (McAfee) were assessed by BSIMM.



**Table 9.1.1 BSIMM V Software Security Framework (SSF)**

|  |  |
| --- | --- |
| **Domain** | **Practice** |
| **Governance** | SM: Strategy and Metrics |
|  | CP: Compliance and Policy |
|  | T: Training |
| **Intelligence** | AM: Attack Models |
|  | SFD: Security Features and Design |
|  | SR: Standards and Requirements |
| **SSDL Touchpoints** | AA: Architecture Analysis |
|  | CR: Code Review |
|  | ST: Security Testing |
| **Deployment** | PT: Penetration Testing |
|  | SE: Software Environment |
|  | CMVM: Configuration Management and Vulnerability Management |

**Table 9.1.2 BSIMM Domains and Practices**

Maturity Levels:

* Level 1: Basic
* Level 2: Typical
* Level 3: Mature

|  |  |
| --- | --- |
| **PSMM** | **BSIMM** |
| **Operational** |  |
| O1 Program | SM1.3: Educate executives.  SM1.6: Require security sign-off.  SM2.3: Create or grow a satellite.  CP2.2: Require security sign-off for compliance-related risk.  CP3.3: Drive feedback from SSDL data back to policy.  SR2.2: Create a standards review board. |
| O2 Resources | SM1.2: Create evangelism role and perform internal marketing.  SR1.2: Create a security portal.  AA2.3: Make SSG available as AA resource or mentor.  AA3.1: Have software architects lead design review efforts. |
| O3 SDL | SM1.4: The software security process will involve release gates/  checkpoints |
| O4 PSIRT | CMVM1.1: Create or interface with incident response.  CMVM1.2: Identify software defects found in operations monitoring and feed them back to development.  CMVM2.1: Have emergency codebase response.  CMVM2.2: Track software bugs found in operations through the fix process.  CMVM3.1: Fix all occurrences of software bugs found in operations.  CMVM3.2: Enhance the SSDL to prevent software bugs found in operations.  CMVM3.3: Simulate software crisis.  CMVM3.4: Operate a bug bounty program. |
| O5 Policy | CP1.3: Create policy.  SR1.3: Translate compliance constraints to requirements.  SR2.5: Create SLA boilerplate. |
| O6 Process | SM1.1: Publish process (roles, responsibilities, plan), evolve as necessary. |
| O7 Training | T1.1: Provide awareness training.  T1.5: Deliver role-specific advanced curriculum (tools, technology stacks, bug parade).  T1.6: Create and use material specific to company history.  T1.7: Deliver on-demand individual training.  T2.5: Enhance satellite through training and events.  T2.6: Include security resources in onboarding.  T2.7: Identify satellite through training.  T3.1: Reward progression through curriculum (certification or HR).  T3.2: Provide training for vendors or outsourced workers.  T3.3: Host external software security events.  T3.4: Require an annual refresher.  T3.5: Establish SSG office hours. |
| O8 Reporting / Tracking Tools | SM2.5: Identify metrics and use them to drive budgets.  SM3.1: Use an internal tracking application with portfolio view.  CP2.3: Implement and track controls for compliance.  CR1.6: Use centralized reporting to close the knowledge loop and drive training.  CMVM2.2: Track software bugs found in operations through the fix process.  CMVM2.3: Develop an operations inventory of applications. |
|  |  |
| **Technical** |  |
| T1 Security Requirements Plan | SFD1.1: Build and publish security features.  SFD1.2: Engage SSG with architecture.  SFD2.1: Build secure-by-design middleware frameworks and common libraries.  SFD2.2: Create SSG capability to solve difficult design problems.  SFD3.1: Form a review board or central committee to approve and maintain secure design patterns.  SFD3.3: Find and publish mature design patterns from the organization. |
| T2 Architecture and Design Reviews | AA1.1: Perform security feature review.  AA1.2: Perform design review for high-risk applications.  AA1.3: Have SSG lead design review efforts.  AA1.4: Use a risk questionnaire to rank applications.  AA2.1: Define and use AA process.  AA2.2: Standardize architectural descriptions (including data flow).  AA3.2: Drive analysis results into standard architecture patterns. |
| T3 Threat Modeling | AM1.1: Build and maintain a top N possible attacks list.  AM1.2: Create a data classification scheme and inventory.  AM1.3: Identify potential attackers.  AM1.4: Collect and publish attack stories.  AM1.5: Gather attack intelligence.  AM1.6: Build an internal forum to discuss attacks.  AM2.1: Build attack patterns and abuse cases tied to potential attackers.  AM2.2: Create technology-specific attack patterns.  AM3.1: Have a science team that develops new attack methods.  AM3.2: Create and use automation to do what attackers will do. |
| T4 Security Testing | ST1.1: Ensure QA supports edge/boundary value condition testing.  ST1.3: Drive tests with security requirements and security features.  ST2.1: Integrate black box security tools into the QA process.  ST2.4: Share security results with QA.  ST3.1: Include security tests in QA automation.  ST3.3: Drive tests with risk analysis results.  ST3.4: Leverage coverage analysis.  ST3.5: Begin to build and apply adversarial security tests (abuse cases). |
| T5 Static Analysis | CR1.4: Use automated tools along with manual review.  CR2.5: Assign tool mentors.  CR2.6: Use automated tools with tailored rules.  CR3.2: Build a factory.  CR3.4: Automate malicious code detection. |
| T6 Dynamic Analysis (Web Apps) | ST2.1: Integrate black box security tools into the QA process.  CR2.5: Assign tool mentors.  CR3.2: Build a factory. |
| T7 Fuzz Testing | ST2.1: Integrate black box security tools into the QA process.  ST3.2: Perform fuzz testing customized to application APIs.  CR2.5: Assign tool mentors. |
| T8 Vulnerability Scans / Penetration Testing | PT1.1: Use external penetration testers to find problems.  PT1.2: Feed results to the defect management and mitigation system.  PT1.3: Use penetration testing tools internally.  PT2.2: Provide penetration testers with all available information.  PT2.3: Schedule periodic penetration tests for application coverage.  PT3.1: Use external penetration testers to perform deep-dive analysis.  PT3.2: Have the SSG customize penetration testing tools and scripts.  CR2.5: Assign tool mentors. |
| T9 Manual Code Reviews | CR1.1: Create a top N bugs list (real data preferred).  CR1.2: Have SSG perform ad hoc review.  CR1.4: Use automated tools along with manual review.  CR1.5: Make code review mandatory for all projects.  CR3.2: Build a factory.  CR3.3: Build a capability for eradicating specific bugs from the entire codebase.  CR3.4: Automate malicious code detection. |
| T10 Secure Coding Standards | SFD3.2: Require use of approved security features and frameworks.  SR1.1: Create security standards.  SR1.4: Use secure coding standards.  SR2.3: Create standards for technology stacks.  CR2.2: Enforce coding standards.  SE2.4: Use code signing.  SE3.2: Use code protection. |
| T11 Open Source / 3rd Party Libraries | CP2.4: Paper all vendor contracts with software security SLAs.  CP3.2: Impose policy on vendors.  SR2.4: Identify open source.  SR3.1: Control open source risk.  SR3.2: Communicate standards to vendors. |
| T12 Privacy | CP1.2: Identify PII obligations.  CP2.1: Identify PII data inventory.  CP2.5: Promote executive awareness of compliance and privacy obligations.  SR2.5: Create SLA boilerplate. |
|  |  |
| **Do Not Map to PSMM** |  |
|  | SM3.2: Run an external marketing program.  CP1.1: Unify regulatory pressures.  SE1.1: Use application input monitoring.  SE1.2: Ensure host and network security basics are in place.  SE2.2: Publish installation guides.  SE3.3: Use application behavior monitoring and diagnostics. |

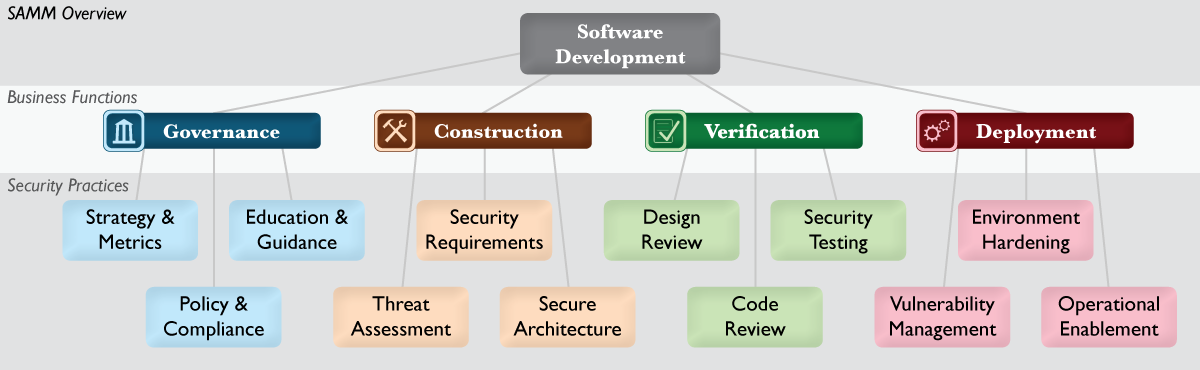
**Table 9.1.3 Mapping of PSMM to BSIMM v5**

## SAMM

**SAMM**: Software Assurance Maturity Model

<http://www.opensamm.org>

Developed by Fortify. Owned and maintained by OWASP.

****

**Diagram 9.2.1 SAMM**

Maturity Levels:

* 0: Implicit starting point with the Practice unfulfilled
* 1: Initial understanding and ad hoc provision of the Practice
* 2: Increase efficiency and/or effectiveness of the Practice
* 3: Comprehensive mastery of the Practice at scale
* +: In-between levels

|  |  |
| --- | --- |
| **PSMM** | **SAMM** |
| **Operational** |  |
| O1 Program | SM1 A. Estimate overall business risk profile  SM1 B. Build and maintain assurance program roadmap  SM2 A. Classify data and applications based on business risk  SM2 B. Establish and measure per-classification security goals |
| O2 Resources | EG2 B. Utilize security coaches to enhance project teams  EG3 A. Create formal application security support portal |
| O3 SDL | DR3 B. Establish release gates for design review  CR3 B. Establish release gates for code review  ST3 B. Establish release gates for security testing |
| O4 PSIRT | VM1 A. Identify point of contact for security issues  VM1 B. Create informal security response team(s)  VM2 A. Establish consistent incident response process  VM2 B. Adopt a security issue disclosure process  VM3 A. Conduct root cause analysis for incidents  VM3 B. Collect per-incident metrics |
| O5 Policy | PC1 A. Identify and monitor external compliance drivers  PC1 B. Build and maintain compliance guidelines  PC2 A. Build policies and standards for security and compliance  PC2 B. Establish project audit practice |
| O6 Process | PC3 A. Create compliance gates for projects  EG1 B. Build and maintain technical guidelines  VM2 A. Establish consistent incident response process  VM2 B. Adopt a security issue disclosure process  OE1 B. Document procedures for typical application alerts |
| O7 Training | EG1 A. Conduct technical security awareness training  EG2 A. Conduct role-specific application security training  EG3 B. Establish role-based examination/certification |
| O8 Reporting / Tracking Tools | SM3 A. Conduct periodic industry-wide cost comparisons  SM3 B. Collect metrics for historic security spend  PC3 B. Adopt solution for audit data collection  VM3 B. Collect per-incident metrics |
|  |  |
| **Technical** |  |
| T1 Security Requirements Plan | SR1 A. Derive security requirements from business functionality  SR1 B. Evaluate security and compliance guidance for requirements  SR2 A. Build an access control matrix for resources and capabilities  SR2 B. Specify security requirements based on known risks  SR3 A. Build security requirements into supplier agreements  SR3 B. Expand audit program for security requirements  DR1 B. Analyze design against known security requirements  CR1 A. Create review checklists from known security requirements |
| T2 Architecture and Design Reviews | SA1 A. Maintain list of recommended software frameworks  SA1 B. Explicitly apply security principles to design  SA2 A. Identify and promote security services and infrastructure  SA2 B. Identify security design patterns from architecture  SA3 A. Establish formal reference architectures and platforms  SA3 B. Validate usage of frameworks, patterns, and platforms  DR1 A. Identify software attack surface  DR1 B. Analyze design against known security requirements  DR2 A. Inspect for complete provision of security mechanisms  DR2 B. Deploy design review service for project teams  DR3 A. Develop data-flow diagrams for sensitive resources  DR3 B. Establish release gates for design review |
| T3 Threat Modeling | TA1 A. Build and maintain application-specific threat models  TA1 B. Develop attacker profile from software architecture  TA2 A. Build and maintain abuse-case models per project  TA2 B. Adopt a weighting system for measurement of threats  TA3 A. Explicitly evaluate risk from third-party components  TA3 B. Elaborate threat models with compensating controls |
| T4 Security Testing | SA3 B. Validate usage of frameworks, patterns, and platforms  ST1 A. Derive test cases from known security requirements  ST2 A. Utilize automated security testing tools  ST2 B. Integrate security testing into development process  ST3 A. Employ application-specific security testing automation  ST3 B. Establish release gates for security testing |
| T5 Static Analysis | CR2 A. Utilize automated code analysis tools  CR2 B. Integrate code analysis into development process  CR3 A. Customize code analysis for application-specific concerns |
| T6 Dynamic Analysis (Web Apps) | <none> |
| T7 Fuzz Testing | <none> |
| T8 Vulnerability Scans / Penetration Testing | ST1 B. Conduct penetration testing on software releases |
| T9 Manual Code Reviews | CR1 B. Perform point-review of high-risk code  CR3 B. Establish release gates for code review |
| T10 Secure Coding Standards | PC2 A. Build policies and standards for security and compliance  OE3 B. Perform code signing for application components |
| T11 Open Source / 3rd Party Libraries | TA3 A. Explicitly evaluate risk from third-party components  SR3 A. Build security requirements into supplier agreements  SA1 A. Maintain list of recommended software frameworks |
| T12 Privacy | <none> |
|  |  |
| **Do Not Map to PSMM** |  |
|  | EH1 A. Maintain operational environment specification  EH1 B. Identify and install critical security upgrades and patches  EH2 A. Establish routine patch management process  EH2 B. Monitor baseline environment configuration status  EH3 A. Identify and deploy relevant operations protection tools  EH3 B. Expand audit program for environment configuration  OE1 A. Capture critical security information for deployment  OE2 A. Create per-release change management procedures  OE2 B. Maintain formal operational security guides  OE3 A. Expand audit program for operational information  OE3 B. Perform code signing for application components |

**Table 9.2.2 Mapping of PSMM to SAMM v1.0**

## DFS

**DFS**: Designed For Security - Intel

<http://sdl.intel.com> >> DFS-Framework.aspx

Developed by Intel to be used internally. Provided certification levels.

The DFS Certification program has been retired as of 2015-02-23.

Design for Security (DFS) is a framework to help organizations prepare for SDL execution and improve their product development capabilities for ensuring that security and privacy objectives are met. Security Champions use the DFS framework as a checklist to ensure that foundational capabilities needed for SDL execution are established and maintained in the organization.

**DFS Levels:**

* **DFS Level 1**: Long term strategic plan focused on resolving the BU business problem in the security assurance space.
* **DFS Level 2**: Demonstrates that the BU is taking the right steps to achieve their final goal of independent, consistent SDL execution.
* **DFS Level 3**: Demonstrates the BU's ability to independently identify and resolve all current and future security assurance issues.
* **DFS Level 4**: Demonstrates the BU's continued improvement efforts, community contribution and leadership in the security assurance space.

|  |  |
| --- | --- |
| **PSMM** | **DFS** |
| **Operational** |  |
| O1 Program | L1: **Process1.2** - BU General manager actively supporting security champion responsibilities.  L4: **Training4.4** - BU participates in cross-org and cross-platform security and privacy technical and BKM sharing forums. |
| O2 Resources | L1: **Process1.1** - BU has assigned security champion to cover all BU.  L1: **Tools1.1** - The BU has identified the appropriate security validation tools needed to address the unique business needs of their product portfolio.  L3: **Process3.5** - BU uses historical resource estimates when creating SDL Plan for technical security activities.  L3: **Tools3.3** - The BU has identified the SW security validation tool needs for their product portfolio.  L3: **Tools3.4** - The BU has integrated and is using the security validation tools identified in the organization's DFS Plan.  L4: **Tools4.1** - BU shares internally developed and externally acquired security tools with Intel security community. |
| O3 SDL | L2 & L3: **SDLExecution2.3** - All SDL non-compliance is approved in Exceptions tool and documented in MySDL. |
| O4 PSIRT | L3: **Process3.3** - Product development teams analyze root cause for Critical & High security and privacy defects and communicates results to responsible parties.  L4: **Validation4.2** - BU communicates all discovered "interdependency" defects in other BU's components to other BU and ensure the defects are addressed (fixed or exception approved) by product ship milestone. |
| O5 Policy | L2: **Process2.3** - All products released by BU are reviewed against the required SDL policies. |
| O6 Process | L2, L3 & L4: **SDLExecution2.1** The BU has implemented processes that ensure all BU projects have closed the assigned SDL activities by close of related SDL milestones on all their SDL projects.  L2, L3 & L4: **SDLExecution2.2** The BU has implemented processes that ensure for all BU projects, milestones are achieved in a timely manner in MySDL and project risks are communicated to management.  L2, L3 & L4: **SDLExecution2.3** The BU has implemented processes that ensure for all BU projects, exceptions for non-compliance with each project’s SDL Plan are filed and closed or approved in a timely manner in MySDL and in Intel’s Exceptions tool.  L3: **Process3.4** - SDL retrospective reviews include an explicit analysis of SDL execution, and includes project security & privacy stakeholders (e.g. SDL Lead, Security Champion, Security Consultant, Privacy Lead).  L4: **Process4.2** - BU has integrated platform SDL process and BKMs. |
| O7 Training | L1: **Training1.1** - BU's security and privacy training plan covers the steps needed to complete all training to achieve the BU's target DFS level.  L1: **Validation1.1** - BU has identified the validation skills necessary to address the security and privacy assurance needs for their product portfolio, and the organization creates training plan and sets aside time to develop the identified skills.  L2 & L3: **Training2.1** - BU has completed 95% of required security training for all its employees.  L2 & L3: **Training2.1** The BU is committed to ensuring that for all existing employees, SDL training trends to 95% completion by end of the year.  L2: **Training2.2** The BU is committed to ensuring that all on-boarded employees (new hires or re-org’d) are assigned required SDL training within three months.  L3: **Training3.1** - Advanced training is completed according to the business needs identified in the BU's DFS Plan.  L4: **Training4.1** - BU contributes to the security and privacy training program (i.e. author, review and teach).  L4: **Training4.2** - BU monitors BU security and privacy training completion to ensure required training is completed by all candidates on timely basis.  L4: **Training4.3** - BU ensures that security engineers maintain security and privacy expertise through continuing education. |
| O8 Reporting / Tracking Tools | L2, L3 & L4: **SDLExecution2.2** - All BU projects are in MySDL and are reporting at least 95% On Track. |
|  |  |
| **Technical** |  |
| T1 Security Requirements Plan | L1: **Process1.3** - BU's DFS plan covers all needed activities to achieve the BU's target DFS level.  L1: **Process1.4** - BU's SDL intercept plan covers all BU projects with details on how and when SDL required activity will be covered.  L2: **Process2.1** - All SDL Milestones & activities assigned to BU projects are included in project planning artifacts [e.g. PLC templates, Project schedule/GANTT chart, PMO documents, PM checklists, MLZ documentation, POP documents]  L2: **Process2.2** - SDL lead for project participates in the end-to-end project planning activity [e.g. MAPP day]  L2: **Process2.5** - All relevant SDL BKM's have been reviewed for inclusion into the BU's development process.  L2: **Process2.6** - All BU project planning milestone reviews include review of SDL requirements (required activities, policies). Project milestone reviews may include POP/PLC milestones or Software milestones.  L2 & L3: **SDLExecution2.1** - All BU projects have closed the assigned SDL activities by the closure of related SDL Milestone.  L3: **Process3.1** - All SDL Activities assigned to the BU's projects are included in project planning.  L4: **Process4.3** - BU uses historical resource estimates when creating SDL Plan for technical security activities. |
| T2 Architecture and Design Reviews | <none> |
| T3 Threat Modeling | <none> |
| T4 Security Testing | L2: **Validation2.1** - All BU MySDL projects include an assigned Security Validation Lead.  L2: **Validation2.2** - All BU security validation plans include test cases for all security objectives and security requirements.  L2: **Validation2.3** - All security tests are executed, and resulting defects dispositioned, by the product ship milestone.  L2: **Validation2.4** - All unfixed Critical and High security vulnerabilities have been documented in approved Exceptions.  L2: **Validation2.2** The BU has implemented processes that ensure that all validation plans include test cases for all security and privacy objectives and requirements.  L3: **Validation3.1** - BU maintains a robust security validation capability and ensures continuity of that capability and skillset.  L3: **Validation3.2** - BU is using the identified security validation tools on all appropriate projects.  L3: **Validation3.4** - SeCoE Detect has performed an in depth review of a project that has gone through a full SDL cycle (or equivalent) and determined that the project demonstrates high quality of execution.  L4: **Validation4.1** - BU identifies the Platform Security Validation Lead, who actively contributes to validation of platforms in which BU's products are included. |
| T5 Static Analysis | L2: **Tools2.1** - BU has assigned ownership of enterprise Static Analysis tools (e.g. Klocwork).  L2: **Tools2.2** - Execution of static analysis software is included in BU standard development process.  L2: **Tools2.3** - BU has integrated ISPO recommended ruleset into integrated Static Analysis tools.  L2: **Tools2.4** - BU performs Static Analysis at every major milestone (alpha, beta, ...) and dispositions all sightings.  L3: **Tools3.1** - BU uses latest released version of Static Analysis tools including IDE-integrated tools.  L3: **Tools3.2** - BU addresses security issues identified by all implemented security tools. |
| T6 Dynamic Analysis (Web Apps) | L3: **Tools3.2** - BU addresses security issues identified by all implemented security tools. |
| T7 Fuzz Testing | L3: **Tools3.2** - BU addresses security issues identified by all implemented security tools.  L4: **Tools4.2** - BU generates and shares custom rules for security tools so that escapes are prevented in the future. |
| T8 Vulnerability Scans / Penetration Testing | L2: **Validation2.4** The BU has implemented processes that ensure that all unfixed Critical/High security/privacy vulnerabilities are filed as exceptions (class = Product Release w/known vulnerabilities) in Intel’s Exceptions tool by product ship milestone.  L3: **Validation3.3** - All Critical and High security vulnerabilities are addressed (fixed or exceptions approved) by product ship milestone. |
| T9 Manual Code Reviews | L2: **Process2.4** - Code review process incorporates SDL Secure Code Checklist. |
| T10 Secure Coding Standards | L4: **Process4.1** - BU has established a standardized set of security acceptance criteria that are enforced for all products received from and exit criteria when delivered to other BU's. |
| T11 Open Source / 3rd Party Libraries | L2: **3rdParty2.1** - All RFP, SOW and contracts created for product procurement must include relevant security and privacy clauses as described on the 3rd Party SDL page  L2: **3rdParty2.2** - All 3rd party components undergo the level of product security and privacy assurance activities [to be completed by Intel, supplier or an independent organization] as deemed necessary after the initial risk assessment. |
| T12 Privacy | L2: **Validation2.3** The BU has implemented processes that ensure that all security and privacy tests are executed, and resulting defects dispositioned, by the product ship milestone.  L3: **Process3.2** - All product changes that go through the BU's change control process must be evaluated for security and privacy risk. |
|  |  |
| **Do Not Map to PSMM** |  |
|  | L3: **Tools3.5** - The BU is using the latest released version of HW security analysis tools on all relevant projects.  L3: **Tools3.6** - Developers address security issues identified by HW security analysis tools.  L3: **Tools3.7** - The BU has identified the HW security validation tool needs for their product portfolio.  L3: **Tools3.8** - The BU has enabled its staff to develop the necessary HW security validation tools |

**Table 9.3.1 Mapping of PSMM to DFS**

## Microsoft SDL Optimized Model

SDL: Security Development Lifecycle

<http://www.microsoft.com/sdl>



**Figure 9.4.1 Microsoft SDL Maturity Levels**

Maturity Levels:

* Level 1: Basic
* Level 2: Standardized
* Level 3: Advanced
* Level 4: Dynamic

## ISO 27034

**ISO/IEC 27034-1:2011(E)**

International Standard ISO/IEC 27034-1

Information technology – Security techniques – Application security, Part 1: Overview and concepts

First edition, 2011-11-15

PSMM parameters are called ASCs in ISO 27034 for Application Security Controls.

|  |
| --- |
| ***8.1.2.6*** The organization should define at least one library of controls for application security. This library is called an Application Security Control Library (ASC Library). It lists and documents all ASCs recognized by the organization. These ASCs evolved from standards, best practices and roles, responsibilities, and professional qualifications, technological, business, and regulatory contexts and application specifications. |

|  |  |
| --- | --- |
| **PSMM** | **ISO 27034-1** |
| **Operational** |  |
| O1 Program | ISO/IEC 27034 |
| O2 Resources | 6.3.10 Roles and permissions  8.1.2.5 ONF - Roles, responsibilities and qualifications  A.8 Roles, Responsibilities and Qualifications  A.9.4 Implementation - 8. Use Approved Tools |
| O3 SDL | 8.1.2.7 Application Security Life Cycle Reference Model  A.2 About the Security Development Lifecycle  A.3 SDL mapped to the Organization Normative Framework |
| O4 PSIRT | ISO 29147 - IT Security - Vulnerability Disclosure  ISO 30111 - IT Security - Vulnerability Handling Processes  6.5.2 Application vulnerabilities  A.9.6 Release - 15. Incident Response Plan |
| O5 Policy | 8.1.2.1 Organization Normative Framework (ONF) – Components – Business context |
| O6 Process | 7.1 Components, processes and frameworks  8.1.2.8 Processes related to application security  8.1.3 Processes related to the Organization Normative Framework  8.3.3 Processes related to the security of the application  8.3.5 Processes involved with the Application Normative Framework  8.4.4 Processes involved with Provisioning and Operating the Application |
| O7 Training | 6.3.4 Application life cycle processes  A.9.1 Training – 1. Training Requirements |
| O8 Reporting / Tracking Tools |  |
|  |  |
| **Technical** |  |
| T1 Security Requirements Plan | 6.3.7 Application specifications  6.4.2 Application security requirements engineering  A.9.2 Requirements – 2. Security Requirements  A.9.6 Release - 16. Final Security Review |
| T2 Architecture and Design Reviews | 8.3 Application Normative Framework  A.9.2 Requirements – 4. Security and Privacy Risk Assessment  A.9.3 Design – 5. Design Requirements |
| T3 Threat Modeling | 8.1.2.4 ONF - Technological context  8.2 Application security risk assessment  A.9.3 Design – 5. Design Requirements  A.9.3 Design – 6. Attack Surface Reduction  A.9.3 Design – 7. Threat Modeling  A.9.5 Verification - 13. Threat Model / Attack Surface Review |
| T4 Security Testing | 8.1.2.6.5.5 ASC Verification Measurement  8.5 Application Security Audit  A.9.2 Requirements – 3. Quality Gates/Bug Bars |
| T5 Static Analysis | A.9.4 Implementation – 10. Static Analysis |
| T6 Dynamic Analysis | A.9.5 Verification – 11. Dynamic Program Analysis |
| T7 Fuzz Testing | A.9.5 Verification – 12. Fuzz Testing |
| T8 Vulnerability Scans / Penetration Testing | 0.4.2 Introduction - Application security is context -dependent – c) technological context  8.1.2.8 Processes related to application security – c)  A.9.2 Requirements - 4. Security and Privacy Risk Assessment – c) |
| T9 Manual Code Reviews | A.9.5 Verification - 14. Manual Code Review (Optional) |
| T10 Secure Coding Standards | 8.1 Organization Normative Framework (ONF)  8.1.2.1 The Business Context – e) best practices for all programming languages |
| T11 Open Source / 3rd Party Libraries | 0.4.2 Introduction - Application security is context -dependent – c) technological context  A.9.6 Release - 15. Incident Response Plan |
| T12 Privacy | 6.3.9 Organization and user data  8.1.2.6.4 Application Level of Trust – Example 2  A.9.2 Requirements - 4. Security and Privacy Risk Assessment: f) (Privacy) Determination of Privacy Impact Rating |

**Table 9.5.1 Mapping of PSMM to ISO 27034**

# Definitions and Abbreviations

**BSIMM** **B**uilding **S**ecurity **i**n **M**aturity **M**odel - Cigital

**BU B**usiness **U**nit

**CSIRT C**omputer **S**ecurity **I**ncident **R**esponse **T**eam

**CVE C**ommon **V**ulnerability **E**numeration

**DAST D**ynamic **A**pplication **S**ecurity **T**esting

**ISecG** **I**ntel **S**ecurity **G**roup

**ISO I**nternational **S**tandards **O**rganization

**M&A M**ergers & **A**cquisitions

**PG P**roduct **G**roup

**PgM P**ro**g**ram **M**anager

**PIA** **P**rivacy **I**mpact **A**ssessment

**PLF** **P**roduct **L**ifecycle **F**ramework

**PM P**roduct **M**anager

**PSC P**roduct **S**ecurity **C**hampion

**PSG P**roduct **S**ecurity **G**roup

**PSIRT P**roduct **S**ecurity **I**ncident **R**esponse **T**eam

**PSMM** **P**roduct **S**ecurity **M**aturity **M**odel - Intel Security

**PT** **P**roduct **T**eam

**SAMM** **S**oftware **A**ssurance **M**aturity **M**odel - OWASP

**SAST S**tatic **A**pplication **S**ecurity **T**esting

**SDET S**oftware **D**eveloper **E**ngineer in **T**est

**SDL S**ecurity **D**evelopment **L**ifecycle

**SDLC S**oftware **D**evelopment **L**ife**c**ycle

**S-PLF** **S**ecurity **P**roduct **L**ifecycle **F**ramework

**SSF** **S**oftware **S**ecurity **F**ramework

|  |  |
| --- | --- |
| R:\Teams\Branding\2014 ISecG\Mar 2014\Logo\Intel_Security_Vertical\Digital_Intel_Security_vrt\int_Security_i_vrt_rgb_3000.png | With its Security Connected strategy, innovative approach to hardware-enhanced security, and unique Global Threat Intelligence, Intel Security is intensely focused on developing proactive, proven security solutions and services that protect systems, networks, and mobile devices for business and personal use around the world. Intel Security combines the experience and expertise of McAfee with the innovation and proven performance of Intel to make security an essential ingredient in every architecture and on every computing platform. Intel Security's mission is to give everyone the confidence to live and work safely and securely in the digital world. [www.intelsecurity.com](http://www.intelsecurity.com).  McAfee, ePolicy Orchestrator, McAfee EMM, VirusScan, SiteAdvisor, and App Alert are trademarks or registered trademarks of McAfee, an Intel Company in the United States and other countries. Other names and brands may be claimed as the property of others.  The product plans, specifications and descriptions herein are provided for information only and subject to change without notice, and are provided without warranty of any kind, express or implied. Copyright © 2016 Intel Security. |
| Intel Security  2200 Mission College Blvd.  Santa Clara, CA 95054-1549  USA  (408) 765-8080  www.intelsecurity.com |

# Document Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| Revision History | | | |
| Name | Version | Change Description | Date |
| Patrick McEnany | 1 | Initial Draft posted on Planet McAfee PSC forum. | 17 Feb 2015 |
| Harold Toomey | 2 | Put into an Intel template Word doc. Organized by parameter instead of by PSMM level. | 14 Apr 2015 |
| Harold Toomey  Patrick McEnany | 3 | Copied text from PSMM slide deck to this doc. Added Tuckman's stages of development. | 5 May 2015 |
| Harold Toomey | 4 | Updated template for Intel | 21 May 2015 |
| Harold Toomey | 5 | Reordered Operational parameters. Removed v1. | 26 May 2016 |
| Harold Toomey | 6 | Synced with XLS updates. Removed periods. | 3 Jun 2015 |
| Harold Toomey | 7 | Rewrote details for Static, Dynamic and Fuzzing sections | 9 Jun 2015 |
| Harold Toomey | 16 | Added to “Security Reviews” and “Vuln Scans”. Added “This parameter measures how well …” to each parameter. | 12 Jun 2015 |
| Harold Toomey | 17 | Added section mapping PSMM to BSIMM and SAMM.  Split out Arch Rev/Threat Model. Combined Dynamic and Fuzz Testing. | 16 Jun 2015 |
| Harold Toomey | 18 | Re-split Dynamic and Fuzz Testing. Removed Certifications and M&A from Operational. Split “Security Requirements and Testing” into two parameters. Added verification section. | 19 Jun 2015 |
| Harold Toomey | 19 | Added comparison charts to DFS and ISO 27034 | 8 July 2015 |
| Harold Toomey | 20 | Rewrote Security Testing descriptions. Added DoD. | 14 Aug 2015 |
| Harold Toomey | 23 | Updated wording and order of several tech parameters. Updated O7-2,3,4; T1-0,3; T3-0,5; T5-0; T12-0,2,3,4,5. Added BSIMM items to all parameters. | 21 Dec 2015 |
| Harold Toomey | 24 | Updated PSIRT order, fixed table numbers. | 29 Feb 2016 |
| Harold Toomey | 25 | Reviewed and updated, especially T4-Fuzz Testing and T5-Vulnerability Scans/ Penetration Testing. | 8 Mar 2016 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |