Secure Software Engineering: Intro

Riccardo Scandariato

Institute of Software Security, TUHH, Germany

ric***do . scanda***to @ tuhh.de

MSc Course Secure Software Engineering – Summer Semester 2022

Learning objectives

- What are ***security*** software processes?

Reading material

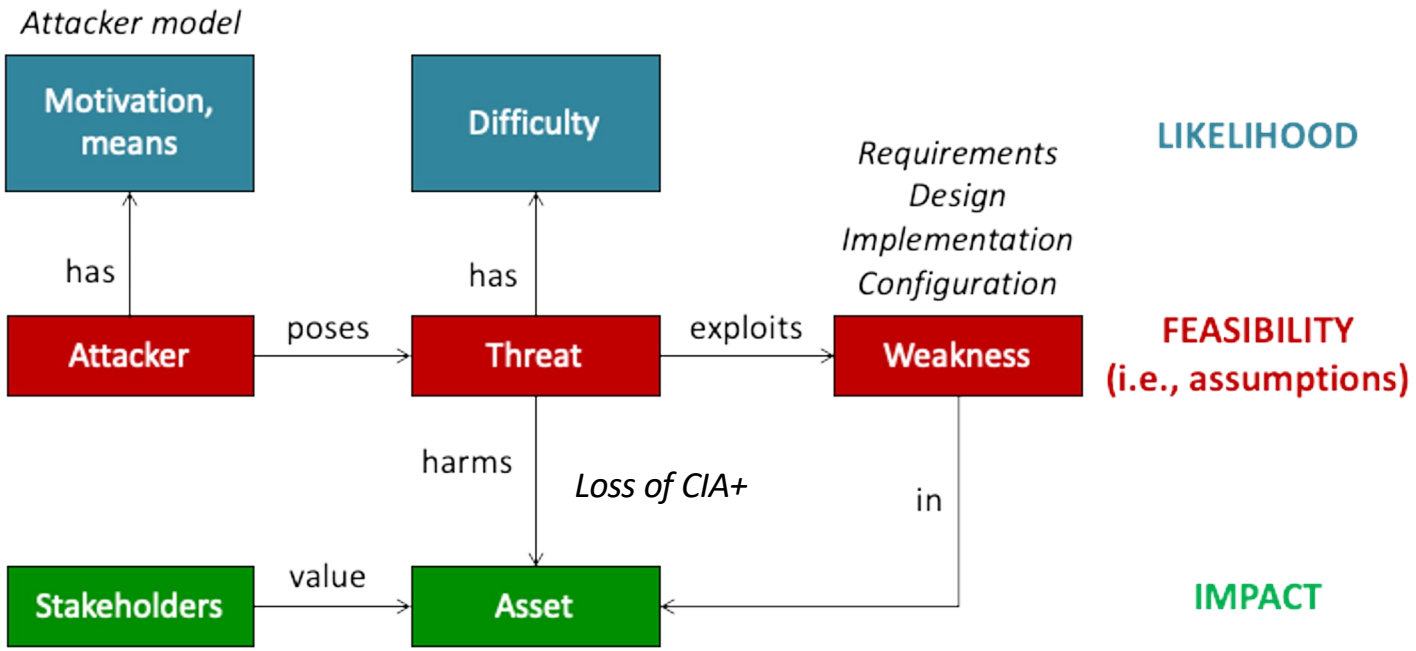
1) https://www.microsoft.com/en-us/securityengineering/sdl/practices

2) B. De Win, et al., **On the secure software development process: CLASP, SDL and Touchpoints compared**, IST, 2009

- What are ***security*** maturity models?

Security flaws

ISO/IEC 15408 (Common Criteria)



CIA+ Confidentiality, Integrity, Availability, Accountability & Non-repudiation, Authenticity

Code-level vulnerabilities

- Examples
  - Stack overflow
  - Command injection
  - SQL injection
  - Cross-site scripting (XSS)
  - Cross-site request forgery (CSRF)
  - …

- MSc course ”Software Security” in Winter Semester

Not just functional code

- Lots of configuration code

  - Container config – Docker, K8s
    (Scanning tools: Aqua, Trivy, Snyk…)

  - Cloud deployment scripts (IaC) – Terraform, Ansible
    (Scanning tools: Snyk, Checov, Terrafirma, TFlint…)

- Configuration vulnerabilites as the new frontier

(cf. A. Rahman et al., The seven sins: security smells in IaC scripts, ICSE 2019)

IaC = Infrastructure-as-Code

Terraform scripting language

```
variable "password" {
  type    = string
  default = "foobarbaz"
}

resource "aws_db_instance" "default" {
  allocated_storage    = 10
  engine               = "mysql"
  engine_version       = "5.7"
  instance_class       = "db.t3.micro"
  name                 = "mydb"
  username             = "foo"
  password             = var.password
  parameter_group_name = "default.mysql5.7"
  skip_final_snapshot  = true
  db_subnet_group_name = aws_db_subnet_group.d
}
```

```
# AWS Secrets Manager
data "aws_secretsmanager_secret" "my_db_secret" {
  name = "my_db_secret"
}

# secret version
data "aws_secretsmanager_secret_version" "my_db_secret" {
  secret_id = data.aws_secretsmanager_secret.my_db_secret.id
}

# store the value in a local variable
locals {
  password = jsondecode(data.aws_secretsmanager_secret_version.my_db_secret.secret_string)
}

# use it like:

resource "aws_db_instance" "default" {
  allocated_storage    = 10
  engine               = "mysql"
  engine_version       = "5.7"
  instance_class       = "db.t3.micro"
  name                 = "mydb"
  username             = "foo"
  password             = local.password
  parameter_group_name = "default.mysql5.7"
  skip_final_snapshot  = true
  db_subnet_group_name = aws_db_subnet_group.default.name
}
```

Above the implementation/ops level

- Is the architectural design right?

- Are the software requirements right?

Architectural security issues

Security weaknesses due to wrong design choices

Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

- Using Weak Authentication (e.g., “API keys”)
- Trust Boundary Violation (e.g., input validation)
- Unprotected Storage of Credentials
- Permission Re-delegation
- …

Upcoming lectures

OWASP Top 10

But also…

- Wrong / missing security assumptions
  - Traffic is not accessible (CAN-bus ?)
  - Traffic is *always* encrypted (until it isn't)
- Wrong perception of risks
  - Low likelihood of malicious internal user
  - Phisical access in IoT systems
- Wrong / missing security requirements
  - In safety-critical systems, ppl focus on integrity over confidentiality
    (what is asset contains personal data?)

PROCESSES

"Shift Left" Paradigm

- Old days: deal with security at the end (patching, incident handling)
- New paradigm, start early in the dev **process**, and deal with security continuously
  - Security-by-design
- Beyond processes and activities? (wait for last slide…)

Secure Development Life-Cycle (SDL)

- Security is not an add-on feature
- You don't write software and then make it "secure" by *adding a few security features* (authentication, etc)
- You don't write software and then make it "secure" by *removing vulnerabilities either* (buffer overflow, etc)
- Security is an ongoing concern throughout the software life cycle
  - Security requirements, Secure design, Secure development – deployment – maintenance (patching, new releases, …)

Microsoft SDL: Historical note

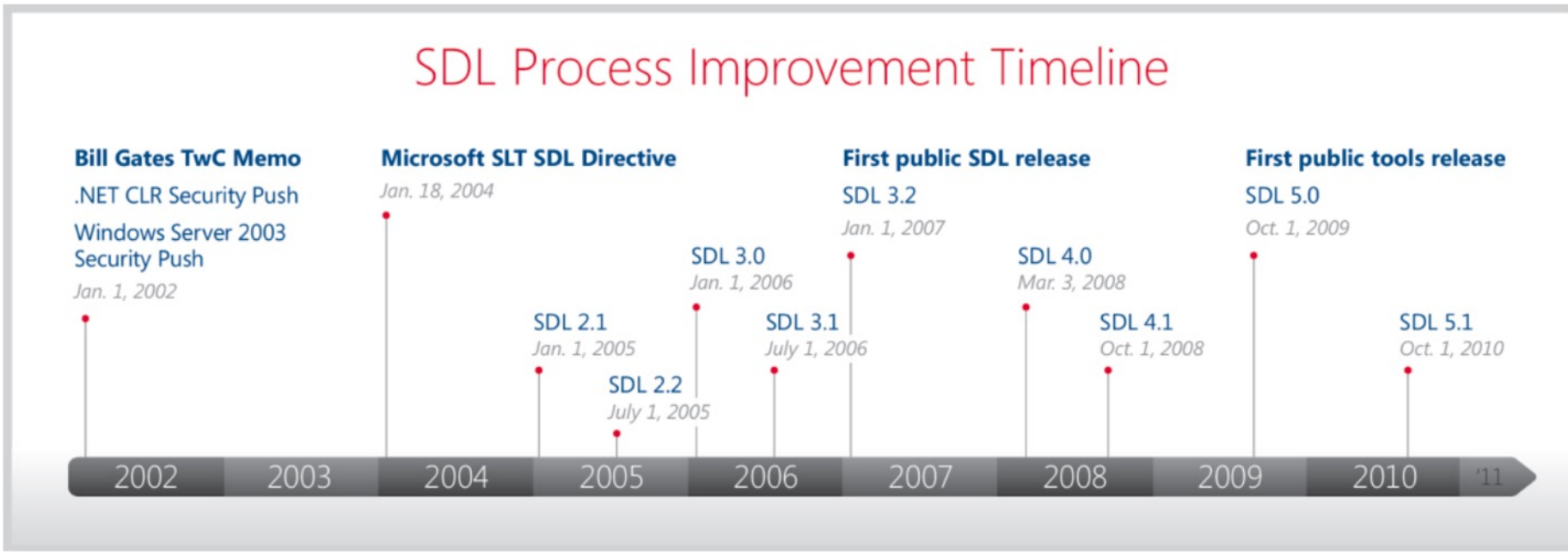
- Bill Gates launched the Trustworthy Computing initiative on January 15, 2002
- Email sent to every full-time employee at Microsoft
- Emphasis on security in the company's strategy
  - "However, even more important than any of these new capabilities is the fact that it is designed from the ground up to deliver Trustworthy Computing."
- Likely, a reaction to the string of malware that had affected Windows (Code Red, Nimda, Slammer...)

https://www.wired.com/2002/01/bill-gates-trustworthy-computing/

Only for your info

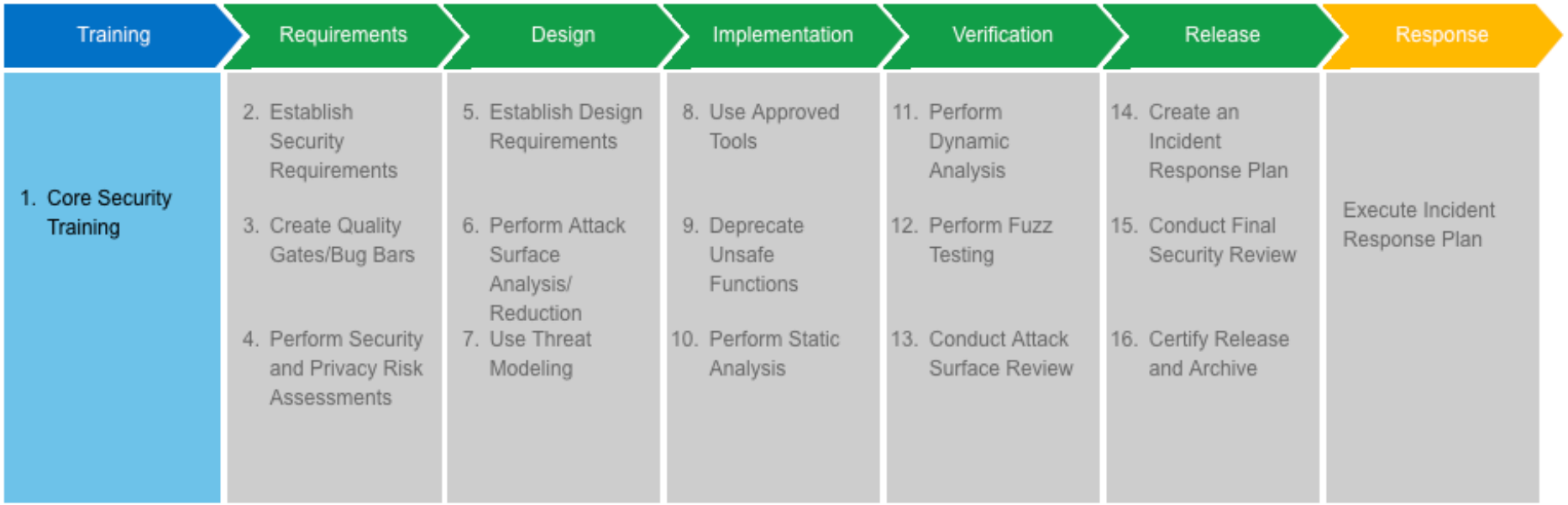
MS SDL evolution

- The SDL Progress Report – Progress reducing software vulnerabilities and developing threat mitigations at Microsoft (2004 – 2010)



Latest version was 5.2 (2012)

Microsoft SDL



Reading material
https://www.microsoft.com/en-us/securityengineering/sdl/practices

Study at home

Example question

- What is a secure software process, in general? What objectives does it pursuit? What activities are central? An example?

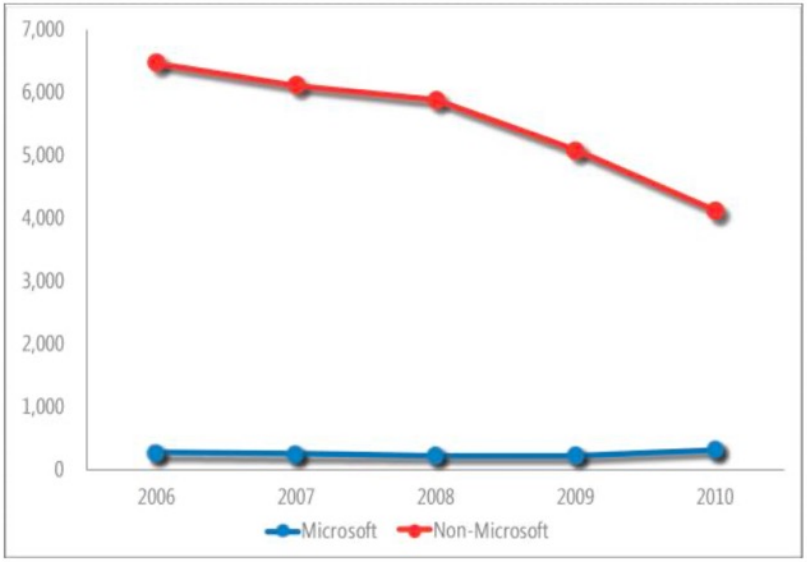
Did it work ?

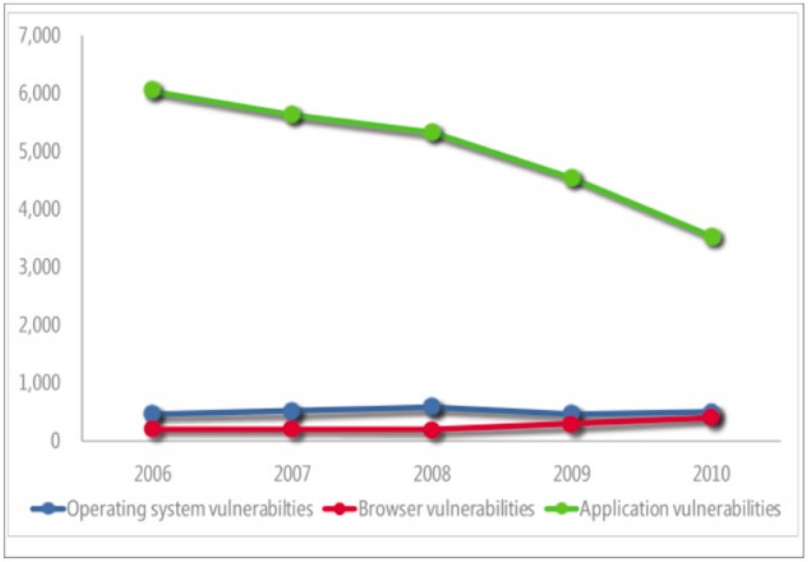
- MS SDL FAQ
- **Has the SDL improved the security of Microsoft products?**
- *As a company-wide initiative and a mandatory policy at Microsoft since 2004, the SDL has played a critical role in embedding security and privacy in Microsoft's culture and software. The SDL has proven to be effective at reducing vulnerability counts of flagship Microsoft products after release.*
  - From "The SDL Progress Report – Progress reducing software vulnerabilities and developing threat mitigations at Microsoft (2004 – 2010)"

Did it work ?

- The SDL Progress Report – Progress reducing software vulnerabilities and developing threat mitigations at Microsoft (2004 – 2010)

Figure 2: *left*: Vulnerability disclosures for Microsoft and non-Microsoft products, 2006 – 2010; *right*: Industry-wide operating system, browser, and application vulnerabilities, 2006 – 2010





Historical note

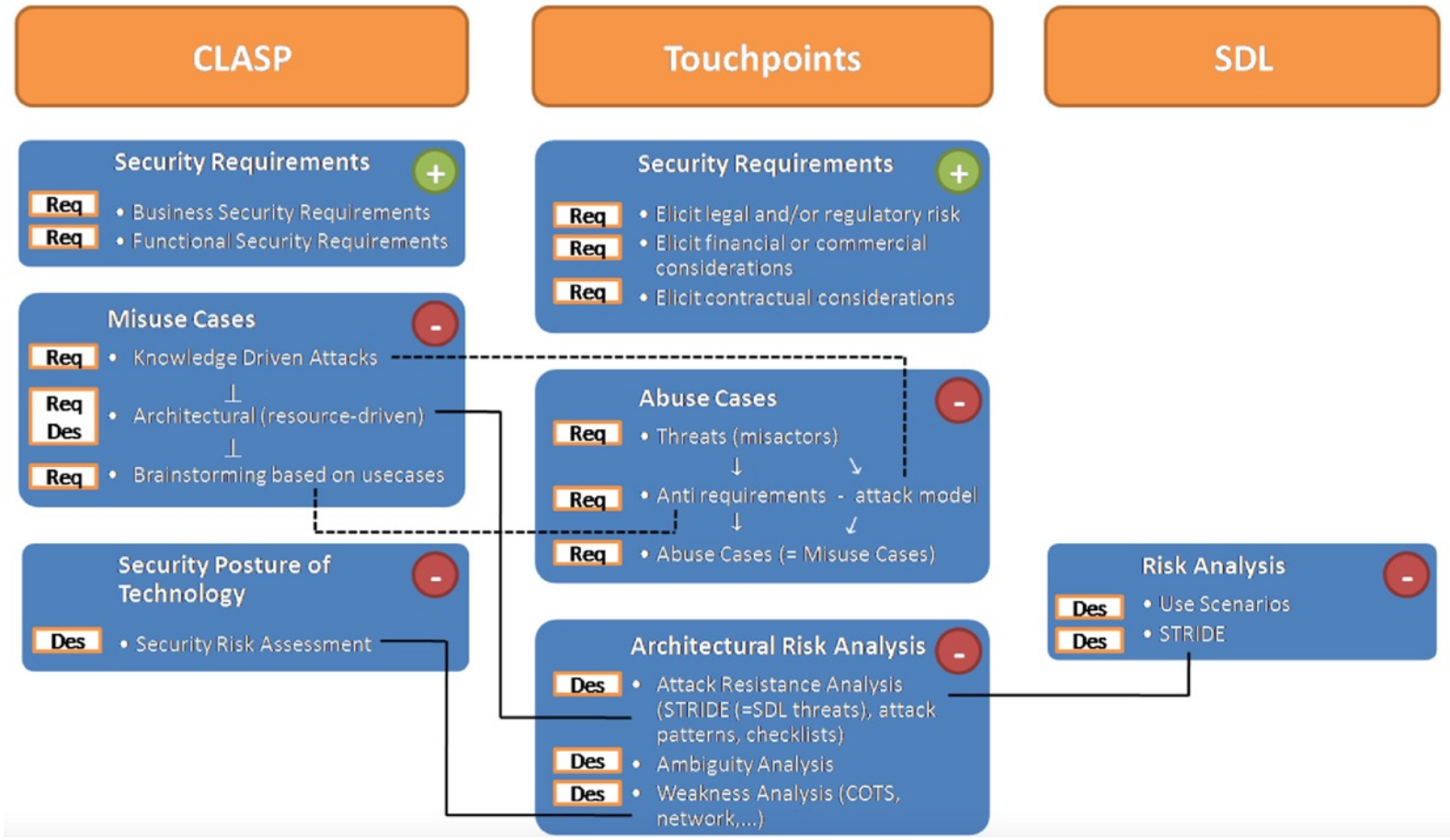
- OWASP CLASP – Comprehensive, Lightweight Application Security Process
  - Many "open" resources provided to incentivize and simplify adoption

- Gary McGraw (famous security consultant) – Touchpoints
  - "Focus on threat modelling and static analysis" (prioritize quick wins)

Reading material
B. De Win, et al., **On the secure software development process: CLASP, SDL and Touchpoints compared**, 2009

CLASP/Touchpoints vs MS SDL

Shift left



Source: Bart De Win (https://owasp.org/www-pdf-archive/OWASPDay2007Belgium_BartDeWin.pdf)

Criticism on secure software processes

- Linear processes are rare (agility)

- Hard to integrate SDL into existing processes
  - Company has to figure it out
  - Some info where made available (limited)

- Compliance requires a big investment, no „growing into it“

Later trend: Maturity Models

- Instead of prescribing a process…
  (sequence of activities and roles who perform them)
- … suggesting areas of intervention and security activities

- McGraw'S Touchpoints → Cigital BSIMM
  - https://www.bsimm.com
- OWASP CLASP → SAMM
  - YouTube channel for SAMM: https://www.youtube.com/channel/UCEZDbvQrj5APg5cEET49A_g
- SDL also evolved towards 12 security practices (simplified SDL)

Maturity models

- Activities can be performed with a varying degree of
  - thoroughness, automation, quality assurance, …
- Company can decide in which area to start / invest more, depending on context, goals, regulatory frameworks, etc.

BSIMM

BSIMM

https://www.bsimm.com

Domains	Practices
Governance	1. Strategy & Metrics (SM) 2. Compliance & Policy (CP) 3. Training (T)
Intelligence	4. Attack Models (AM) 5. Security Features & Design (SFD) 6. Standards & Requirements (SR)
SSDL Touchpoints	7. Architecture Analysis (AA) 8. Code Review (CR) 9. Security Testing (ST)
Deployment	10. Penetration Testing (PT) 11. Software Environment (SE) 12. Configuration Management & Vulnerability Management (CMVM)

Meaning of “Levels of assurance”

- More activities, more thoroughly

SSDL Touchpoints

ARCHITECTURE ANALYSIS (AA)		
LEVEL 1		
ACTIVITY DESCRIPTION	ACTIVITY #	PARTICIPANT %
Perform security feature review.	AA1.1	86%
Perform design review for high-risk applications.	AA1.2	37%
Have SSG lead design review efforts.	AA1.3	28%
Use a risk questionnaire to rank applications.	AA1.4	59%
LEVEL 2		
Define and use AA process.	AA2.1	15%
Standardize architectural descriptions (including data flow).	AA2.2	12%
Make SSG available as AA resource or mentor.	AA2.3	17%
LEVEL 3		
Have software architects lead design review efforts.	AA3.1	8%
Drive analysis results into standard architecture patterns.	AA3.2	1%

Domain: Governance

Practices help to organise, manage and measure a Software Security Initiatives (SII)

- Strategy & Metrics
- Compliance & Policy
- Training

Only for your info

GOVERNANCE		
STRATEGY & METRICS (SM)	**COMPLIANCE & POLICY (CP)**	**TRAINING (T)**
LEVEL 1	**LEVEL 1**	**LEVEL 1**
• [SM1.1] Publish process and evolve as necessary. • [SM1.3] Educate executives on software security. • [SM1.4] Implement lifecycle instrumentation and use to define governance.	• [CP1.1] Unify regulatory pressures. • [CP1.2] Identify PII obligations. • [CP1.3] Create policy.	• [T1.1] Conduct software security awareness training. • [T1.7] Deliver on-demand individual training. • [T1.8] Include security resources in onboarding.
LEVEL 2	**LEVEL 2**	**LEVEL 2**
• [SM2.1] Publish data about software security internally and drive change. • [SM2.2] Verify release conditions with measurements and track exceptions. • [SM2.3] Create or grow a satellite. • [SM2.6] Require security sign-off prior to software release. • [SM2.7] Create evangelism role and perform internal marketing.	• [CP2.1] Build PII inventory. • [CP2.2] Require security sign-off for compliance-related risk. • [CP2.3] Implement and track controls for compliance. • [CP2.4] Include software security SLAs in all vendor contracts. • [CP2.5] Ensure executive awareness of compliance and privacy obligations.	• [T2.5] Enhance satellite through training and events. • [T2.8] Create and use material specific to company history. • [T2.9] Deliver role-specific advanced curriculum.
LEVEL 3	**LEVEL 3**	**LEVEL 3**
• [SM3.1] Use an internal tracking application with portfolio view. • [SM3.2] SSI efforts are part ofexternal marketing. • [SM3.3] Identify metrics and use them to drive resourcing. • [SM3.4] Integrate software-defined lifecycle governance.	• [CP3.1] Create a regulator compliance story. • [CP3.2] Impose policy on vendors. • [CP3.3] Drive feedback from software lifecycle data back to policy.	• [T3.1] Reward progression through curriculum. • [T3.2] Provide training for vendors and outsourced workers. • [T3.3] Host software security events. • [T3.4] Require an annual refresher. • [T3.5] Establish SSG office hours. • [T3.6] Identify new satellite members through observation.

Domain: Intelligence

Practices result in collection and identification of corporate intelligence related with SSI

- Attack Models
- Security Features & Design
- Standards & Requirements

Only for your info

INTELLIGENCE		
ATTACK MODELS (AM)	**SECURITY FEATURES & DESIGN (SFD)**	**STANDARDS & REQUIREMENTS (SR)**
LEVEL 1	**LEVEL 1**	**LEVEL 1**
• [AM1.2] Create a data classification scheme and inventory. • [AM1.3] Identify potential attackers. • [AM1.5] Gather and use attack intelligence.	• [SFD1.1] Integrate and deliver security features. • [SFD1.2] Engage the SSG with architecture teams.	• [SR1.1] Create security standards. • [SR1.2] Create a security portal. • [SR1.3] Translate compliance constraints to requirements.
LEVEL 2	**LEVEL 2**	**LEVEL 2**
• [AM2.1] Build attack patterns and abuse cases tied to potential attackers. • [AM2.2] Create technology-specific attack patterns. • [AM2.5] Maintain and use a top *N* possible attacks list. • [AM2.6] Collect and publish attack stories. • [AM2.7] Build an internal forum to discuss attacks.	• [SFD2.1] Leverage secure-by-design components and services. • [SFD2.2] Create capability to solve difficult design problems.	• [SR2.2] Create a standards review board. • [SR2.4] Identify open source. • [SR2.5] Create SLA boilerplate.
LEVEL 3	**LEVEL 3**	**LEVEL 3**
• [AM3.1] Have a research group that develops new attack methods. • [AM3.2] Create and use automation to mimic attackers. • [AM3.3] Monitor automated asset creation.	• [SFD3.1] Form a review board or central committee to approve and maintain secure design patterns. • [SFD3.2] Require use of approved security features and frameworks. • [SFD3.3] Find and publish secure design patterns from the organization.	• [SR3.1] Control open source risk. • [SR3.2] Communicate standards to vendors. • [SR3.3] Use secure coding standards. • [SR3.4] Create standards for technology stacks.

Domain: SSDL Touchpoints

Essential security best practices required in Software development phases (SDLC)

- Architecture Analysis
- Code Review
- Security Testing

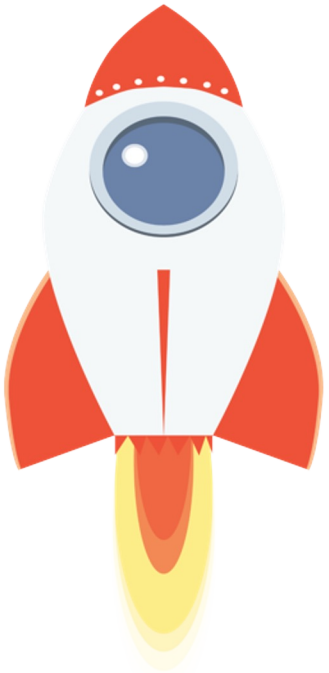
Only for your info

SSDL TOUCHPOINTS		
ARCHITECTURE ANALYSIS (AA)	**CODE REVIEW (CR)**	**SECURITY TESTING (ST)**
LEVEL 1	**LEVEL 1**	**LEVEL 1**
• [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 methodology to rank applications.	• [CR1.2] Perform opportunistic code review. • [CR1.4] Use automated tools. • [CR1.5] Make code review mandatory for all projects. • [CR1.6] Use centralized reporting to close the knowledge loop. • [CR1.7] Assign tool mentors.	• [ST1.1] Ensure QA performs edge/boundary value condition testing. • [ST1.3] Drive tests with security requirements and security features. • [ST1.4] Integrate opaque-box security tools into the QA process.
LEVEL 2	**LEVEL 2**	**LEVEL 2**
• [AA2.1] Define and use AA process. • [AA2.2] Standardize architectural descriptions.	• [CR2.6] Use automated tools with tailored rules. • [CR2.7] Use a top *N* bugs list (real data preferred).	• [ST2.4] Share security results with QA. • [ST2.5] Include security tests in QA automation. • [ST2.6] Perform fuzz testing customized to application APIs.
LEVEL 3	**LEVEL 3**	**LEVEL 3**
• [AA3.1] Have engineering teams lead AA process. • [AA3.2] Drive analysis results into standard design patterns. • [AA3.3] Make the SSG available as an AA resource or mentor.	• [CR3.2] Build a capability to combine assessment results. • [CR3.3] Create capability to eradicate bugs. • [CR3.4] Automate malicious code detection. • [CR3.5] Enforce coding standards.	• [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). • [ST3.6] Implement event-driven security testing in automation.

Domain: Deployment

Practices that deals with network security and software maintenance requirements

- Penetration Testing
- Software Environments
- Configuration Management & Vulnerability Management



Only for your info

DEPLOYMENT

PENETRATION TESTING (PT)	SOFTWARE ENVIRONMENT (SE)	CONFIGURATION MANAGEMENT & VULNERABILITY MANAGEMENT (CMVM)
LEVEL 1	**LEVEL 1**	**LEVEL 1**
• [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.	• [SE1.1] Use application input monitoring. • [SE1.2] Ensure host and network security basics are in place.	• [CMVM1.1] Create or interface with incident response. • [CMVM1.2] Identify software defects found in operations monitoring and feed them back to development.
LEVEL 2	**LEVEL 2**	**LEVEL 2**
• [PT2.2] Penetration testers use all available information. • [PT2.3] Schedule periodic penetration tests for application coverage.	• [SE2.2] Define secure deployment parameters and configurations. • [SE2.4] Protect code integrity. • [SE2.5] Use application containers to support security goals. • [SE2.6] Ensure cloud security basics. • [SE2.7] Use orchestration for containers and virtualized environments.	• [CMVM2.1] Have emergency response. • [CMVM2.2] Track software bugs found in operations through the fix process. • [CMVM2.3] Develop an operations inventory of software delivery value streams.
LEVEL 3	**LEVEL 3**	**LEVEL 3**
• [PT3.1] Use external penetration testers to perform deep-dive analysis. • [PT3.2] Customize penetration testing tools.	• [SE3.2] Use code protection. • [SE3.3] Use application behavior monitoring and diagnostics. • [SE3.6] Enhance application inventory with operations bill of materials.	• [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 crises. • [CMVM3.4] Operate a bug bounty program. • [CMVM3.5] Automate verification of operational infrastructure security. • [CMVM3.6] Publish risk data for deployable artifacts. • [CMVM3.7] Streamline incoming responsible vulnerability disclosure.

Study at home

Example question

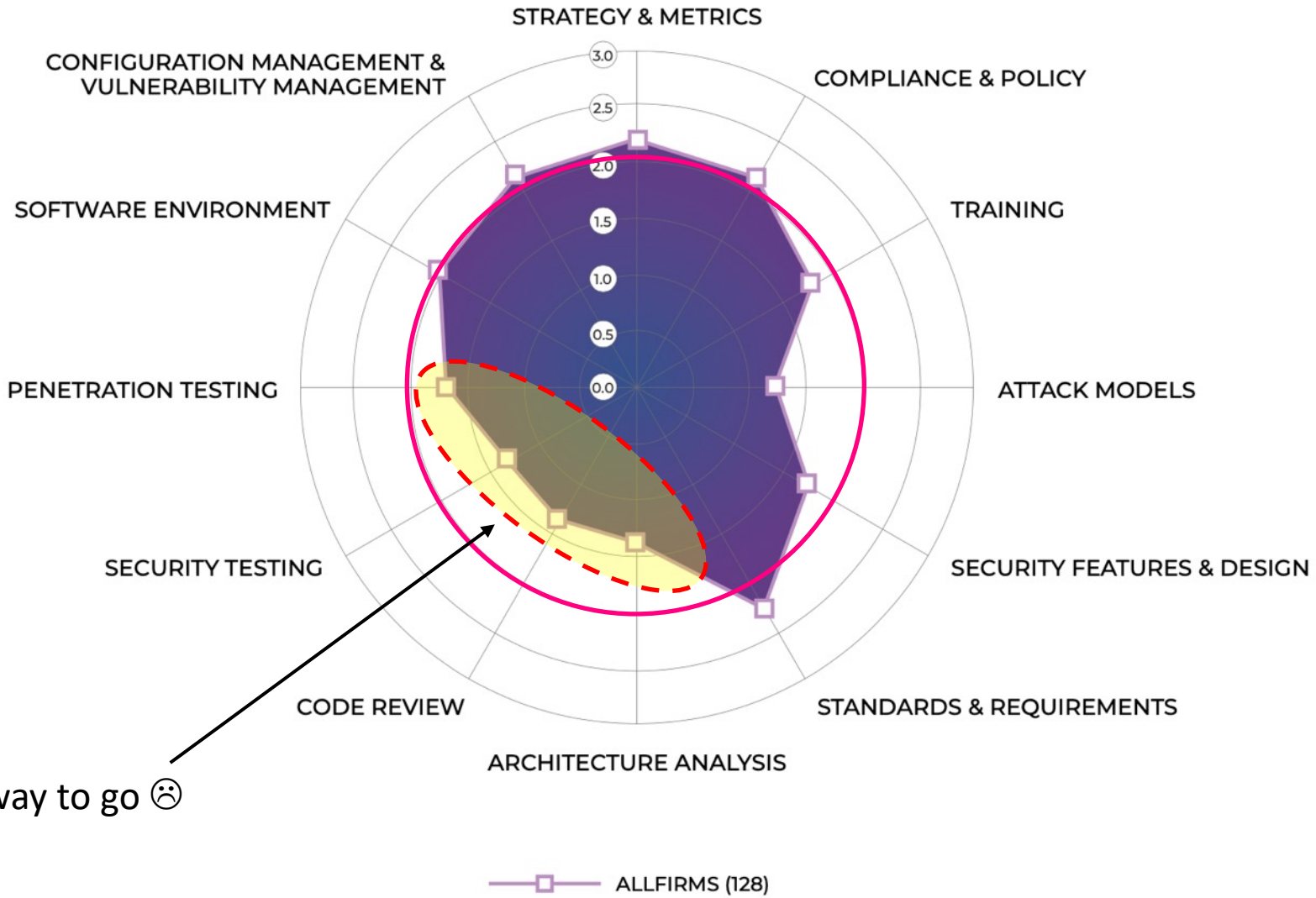
- Difference in the practices described by SDL and BSIMM? Philosophical difference between SDL and BSIMM?

Benchmarking

- Data from 128 of companies
  - Not disclosed !
  - Not independently validated !

- SAMM is creating a similar initiative https://owaspsamm.org/benchmarking/

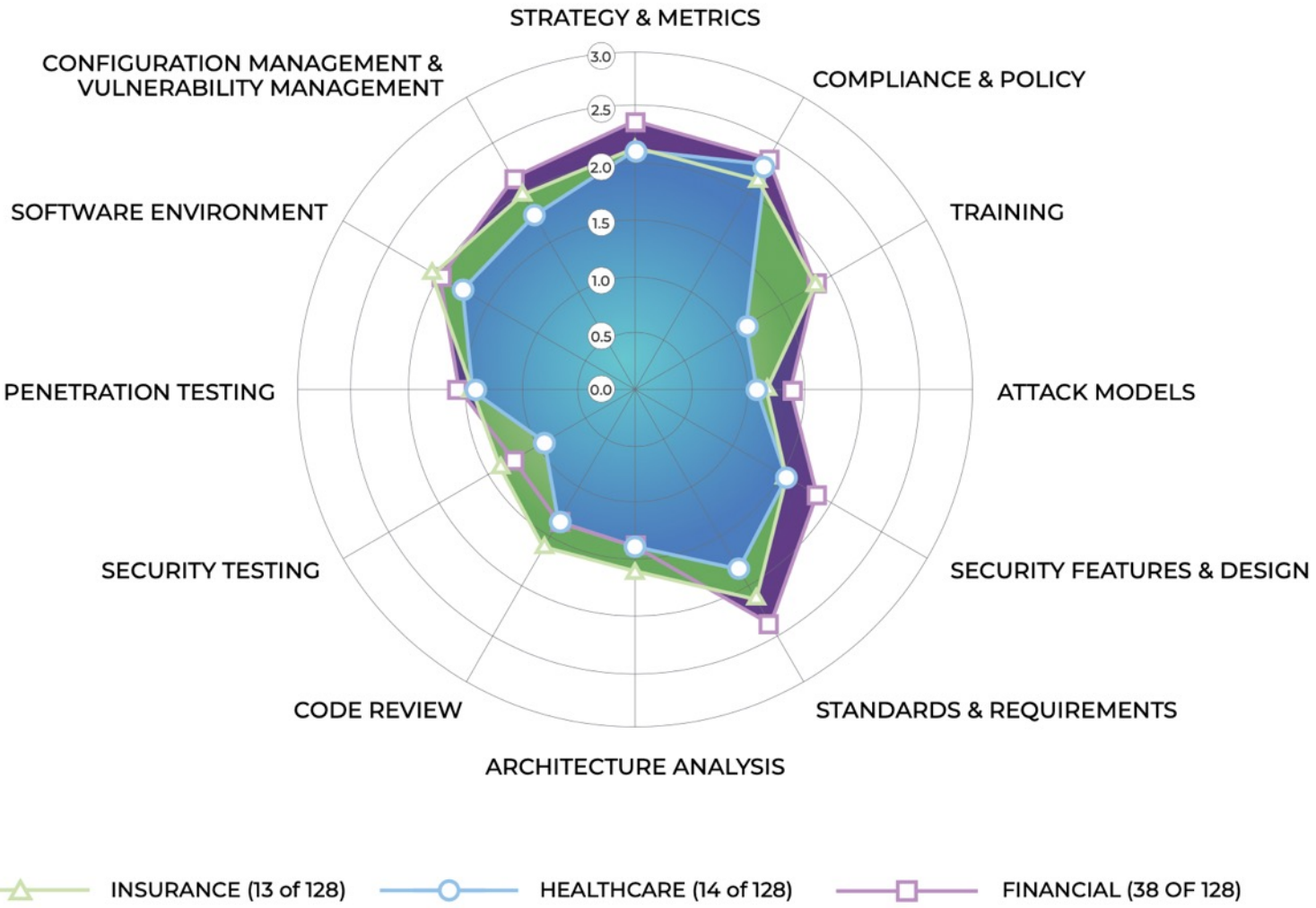
Benchmarking (all companies)



Long way to go ☹

Study at home

Benchmarking (per domain)



Only for your info

SAMM

BSIMM

Governance	Design
Strategy and Metrics	Threat Assessment
Policy and Compliance	Security Requirements
Education and Guidance	Security Architecture

Implementation	Verification	Operations
Secure Build	Architecture Assessment	Incident Management
Secure Deployment	Requirements-driven Testing	Environment Management
Defect Management	Security Testing	Operational Management

Beyond processes

- Prior idea: "certifying" the process, as a way to provide security assurance
  - *Core Infreastructure Initiative (CII*)* follows the same principle but makes it more lightweight (badges for best practices)
  - Badges: Passing, Silver, Gold
  - The security part of the badge is rather product-focussed
  - https://bestpractices.coreinfrastructure.org/en/criteria/0

- Different focus: product/service and its development artifacts
  - Common Criteria had this right ;)

- Security Assurance Cases
  - Product-focuss and evidence-based

*CII now replaced by OpenSSF (Open Source Security Foundation)

Latest trend: certification

- Regulation (EU) 2019/881 (Cybersecurity Act)
  - EU Cybersecurity Certification Framework

- EU Cybersecurity Certification scheme
  - Developed by ENISA (UE Agency for Cybersecurity)
  - Based on Common Criteria (and ISO standards)
  - Version V.1.1.1 (candidate)
  - 300 pages ☺

Learning objectives: checkpoint

- What are ***security*** software processes?
  - *Set of additional security-centric activities that are added to the phases of the software process (plus focus on training)*
- What are ***security*** maturity models?
  - *Areas of intervention that describe increasingly more thorough security activities to be carried out in an organization (focus on software development)*