Cost of Developing Secure Software

Elaine Venson

Brad Clark

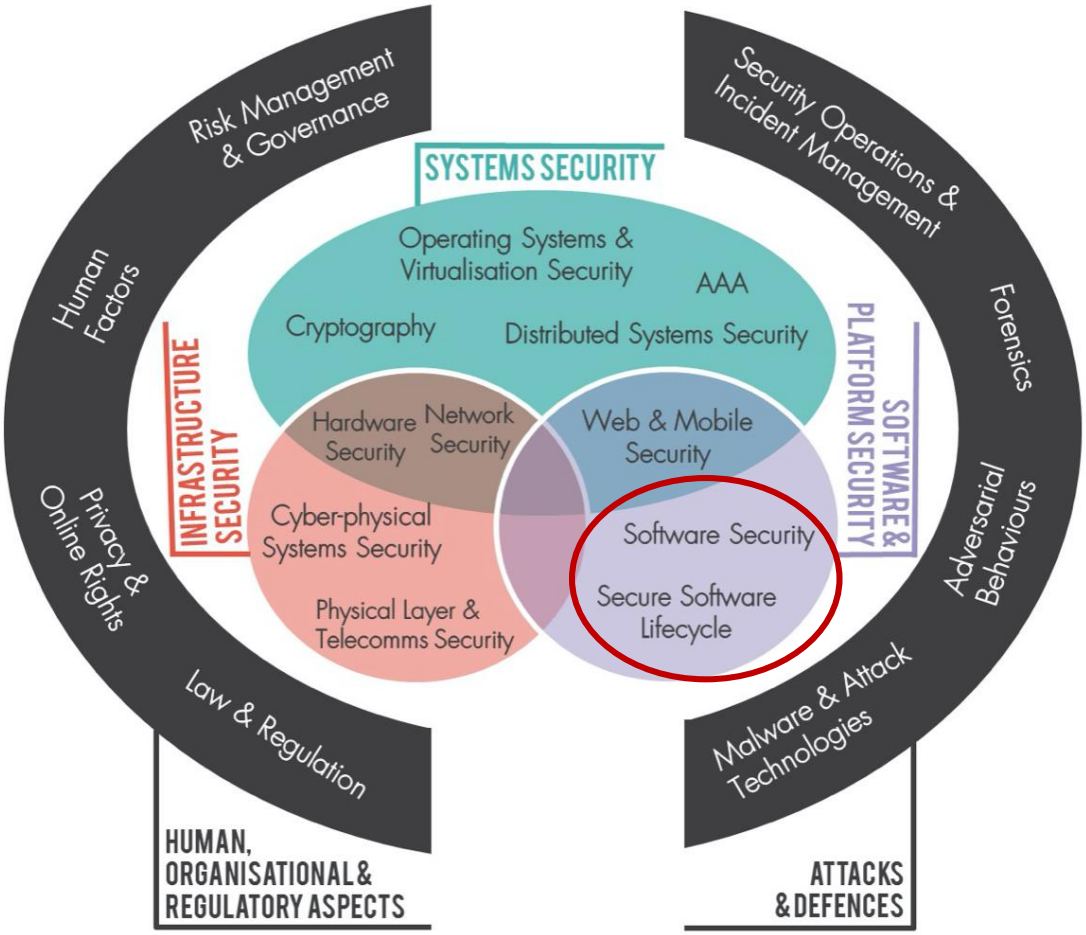
Barry Boehm

September 16, 2020

Outline

- ➢Background
- Previous studies on secure software development costs
- Proposed rating scale
- Data collection and Initial results of a Wide-band Delphi
- Next steps

CyBoK



Software Security

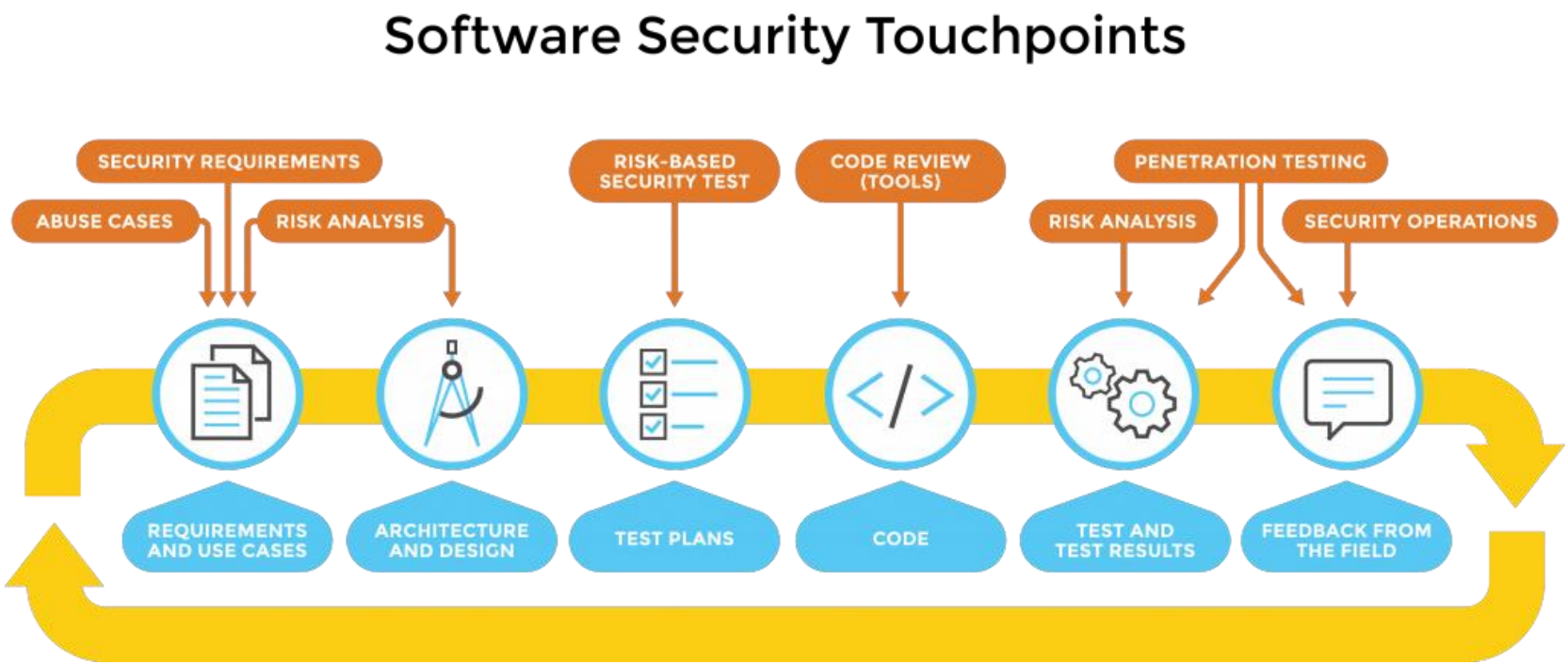
Known **categories of programming errors resulting in security bugs, & techniques for avoiding these errors**—both through coding practice and improved language design—and tools, techniques, and methods for detection of such errors in existing systems.

Secure Software Lifecycle

The **application of security software engineering techniques** in the whole systems development lifecycle resulting in software that is secure by default.

Source: https://www.cybok.org/

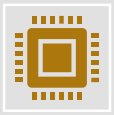
Secure Software Development (Touchpoints)



https://www.slideshare.net/blackducksoftware/managing-open-source-in-application-security-and-software-development-lifecycle

Software Security

Engineering software that continues working under malicious attack [McGraw, 2006]



Many issues faced in computer security today are rooted in our approach to developing software and systems [Heitzenrater, 2016]

Software defects have security ramifications [McGraw, 2006]

Security is an emergent property of a software system. There is no single addition of a feature that can make a software secure [McGraw, 2006]

Security Objectives and Requirements

- Security objectives establish *confidentiality*, *integrity* and *availability* requirements
- Functional
  - Security Features
  - Security Objectives -> Security Functional Components (CC)
- Non-functional (a quality requirement)
- In practice most software systems do not have explicit security objective/requirement
- Software security if often about *avoiding known classes of bugs* that enable attacks

Outline

- Background

➢Previous studies on secure software development costs

- Proposed rating scale
- Data collection and Initial results of a Wide-band Delphi
- Next steps

Sources of Cost (extracted from literature)

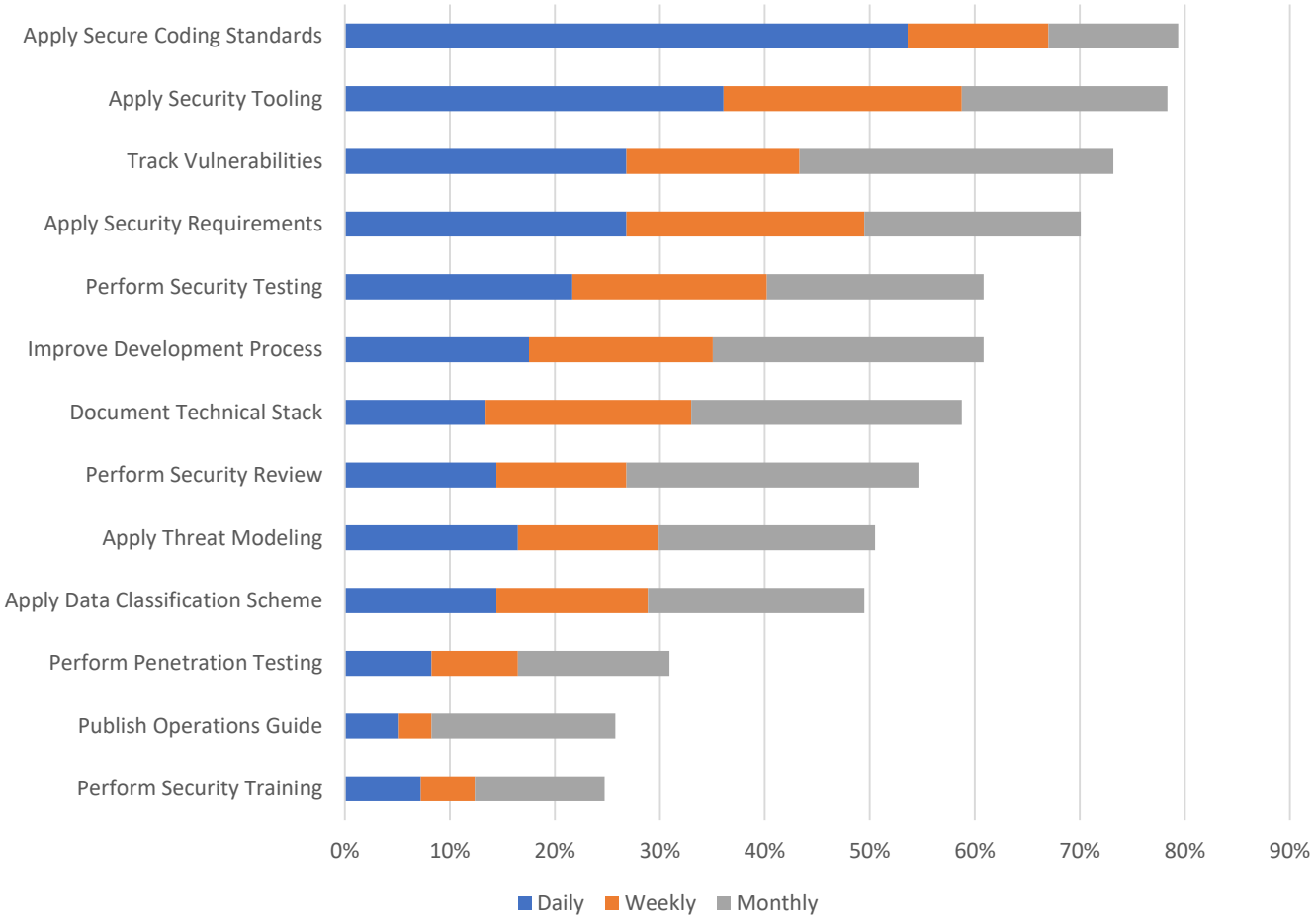
Source	Papers	Source	Papers
Perform Security Review	21	Perform Security Training	6
Apply Threat Modeling	18	Improve Development Process	5
Perform Security Testing	16	Perform Penetration Testing	5
Apply Security Requirements	11	Achieve Security Level	3
Apply Security Tooling	11	Document Technical Stack	3
Implement Countermeasures	9	Security Experts, Security Groups, Security Master	3
Fix Vulnerabilities	9	Track Vulnerabilities	3
Apply Secure Coding Standards	8	Functional Features	2
Apply Data Classifications Scheme	7	Hardening Procedures	2
Publish Operations Guide	7	Security by Design Paradigm	1

SWSec Practices Other Sources

Approaches to Estimating Costs of SWSec

Approach	Additional Cost	Productivity Range	Source	Validation
COCOMO II security extension	0.94 (Low) 1.02 (Nominal) 1.27 (High) 1.43 (Very High) 1.75 (Extra High)	1.86	Expert estimation	Not validated
COSECMO	0% (Nominal) 20% to 80% (EAL 3 - High) 50 to 200% (EAL 4 - Very High) 125% to 500% (EAL 5 - Extra High) 313% to 1250% (EAL 6 - Super High) 781% to 3125% (EAL 7 - Ultra High)	31.25	Expert estimation with two inputs provided by a Commercial Company	Not validated
Weapon systems development cost model (COCOMO II based)	1.0 (Low or Nominal) 1.87 (High)	1.87	Expert estimation and 73 data points	Cross validation
Secure OS software cost model (COCOMO II based)	1 (Nominal) 1.25 to 1.5 (High) 1.75 to 2.0 (Very High) 2.0 to 2.75 (Extra High) 3.0 to 3.75 (Super High)	3.75	Expert estimation	Case study
FPA security extension (from General System Characteristics)	0 to 5% increase in the function points size of the project	1.05	Practices from survey with developers	Not validated

Practices Usage (from practitioners)



State of Art versus Practice

- Sources of cost
  - Practices found in literature are applied in industry, i.e., literature and survey are consistent
- Estimation methods
  - Academia: COCOMO-based models
  - Industry: expert estimation
- Added costs for security
  - For rating = High:
    - COCOMO-based models -> 20% to 80% added effort
    - Survey results for New Development projects -> 31.6% average, 25% median (risk level 4)
      - Level 1: low likelihood of attack and low impact of attack
      - Level 5: high likelihood and high impact of attack

Outline

- Background
- Previous studies on secure software development costs
- ➢Proposed rating scale
- Data collection and Initial results of a Wide-band Delphi
- Next steps

Security Driver for COCOMO III

- Based on software security practices
  - Found in literature
  - Usage confirmed in survey
- Degrees of applying security practices based on existing studies
  - Building Security In Maturity Model (BSIMM) v10
    - Published in 2019 with data observed form 122 firms
  - OWASP Software Assurance Maturity Model (SAMM) v1.5
    - An open framework to help organizations formulate and implement a strategy for software security
    - Published in 2017 with contributions from industry
- Rating scale will provide a measure of the level of secure software development

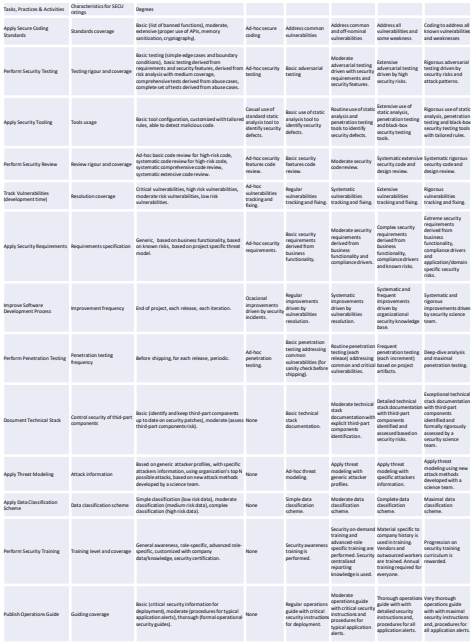
Practices Consolidation

Practices + BSIMM + SAMM =

Tasks, Practices & Activities	Characteristics for SECU ratings	Degrees					
Apply Secure Coding Standards	Standards coverage	Basic (list of banned functions), moderate, extensive (proper use of APIs, memory sanitization, cryptography).	Ad-hoc secure coding	Address common vulnerabilities	Address common and off-nominal vulnerabilities	Address all vulnerabilities and some weakness	Coding to address all known vulnerabilities and weaknesses
Perform Security Testing	Testing rigour and coverage	Basic testing (simple edge cases and boundary conditions), basic testing derived from requirements and security features, derived from risk analysis with medium coverage, comprehensive tests derived from abuse cases, complete set of tests derived from abuse cases.	Ad-hoc security testing	Basic adversarial testing	Moderate adversarial testing driven with security requirements and security features.	Extensive adversarial testing driven by high security risks.	Rigorous adversarial testing driven by security risks and attack patterns.
Apply Security Tooling	Tools usage	Basic tool configuration, customized with tailored rules, able to detect malicious code.	Casual use of standard static analysis tool to identify security defects.	Basic use of static analysis tool to identify security defects.	Routine use of static analysis and penetration testing tools to identify security defects.	Extensive use of static analysis, penetration testing and black-box security testing tools.	Rigorous use of static analysis, penetration testing and black-box security testing tools with tailored rules.
Perform Security Review	Review rigour and coverage	Ad-hoc basic code review for high-risk code, systematic code review for high-risk code, systematic comprehensive code review, systematic extensive code review.	Ad-hoc security features code review.	Basic security features code review.	Moderate security code review.	Systematic extensive security code and design review.	Systematic rigorous security code and design review.
Track Vulnerabilities (development time)	Resolution coverage	Critical vulnerabilities, high risk vulnerabilities, moderate risk vulnerabilities, low risk vulnerabilities.	Ad-hoc vulnerabilities tracking and fixing.	Regular vulnerabilities tracking and fixing.	Systematic vulnerabilities tracking and fixing.	Extensive vulnerabilities tracking and fixing.	Rigorous vulnerabilities tracking and fixing.
Apply Security Requirements	Requirements specification	Generic, based on business functionality, based on known risks, based on project specific threat model.	Ad-hoc security requirements.	Basic security requirements derived from business functionality.	Moderate security requirements derived from business functionality and compliance drivers.	Complex security requirements derived from business functionality, compliance drivers and known risks.	Extreme security requirements derived from business functionality, compliance drivers and application/domain specific security risks.
Improve Software Development Process	Improvement frequency	End of project, each release, each iteration.	Ocasional improvements driven by security incidents.	Regular improvements driven by vulnerabilities resolution.	Systematic improvements driven by vulnerabilities resolution.	Systematic and frequent improvements driven by organizational security knowledge base.	Systematic and rigorous improvements driven by security science team.
Perform Penetration Testing	Penetration testing frequency	Before shipping, for each release, periodic.	Ad-hoc penetration testing.	Basic penetration testing addressing common vulnerabilities (for sanity check before shipping).	Routine penetration testing (each release) addressing common and critical vulnerabilities.	Frequent penetration testing (each increment) based on project artifacts.	Deep-dive analysis and maximal penetration testing.
Document Technical Stack	Control security of thid-part components	Basic (identify and keep third-part components up to date on security patches), moderate (assess third-part components risk).	None	Basic technical stack documentation.	Moderate technical stack documentation with explicit third-part components identification.	Detailed technical stack documentation with third-part components identified and assessed based on security risks.	Exceptional technical stack documentation with third-part components identified and formally rigorously assessed by a security science team.
Apply Threat Modeling	Attack information	Based on generic attacker profiles, with specific attackers information, using organization's top N possible attacks, based on new attack methods developed by a science team.	None	Ad-hoc threat modeling.	Apply threat modeling with generic attacker profiles.	Apply threat modeling with specific attackers information.	Apply threat modeling using new attack methods developed with a science team.
Apply Data Classification Scheme	Data classification scheme	Simple classification (low risk data), moderate classification (medium risk data), complex classification (high risk data).	None	Simple data classification scheme.	Moderate data classification scheme.	Complete data classification scheme.	Maximal data classification scheme.
Perform Security Training	Training level and coverage	General awareness, role-specific, advanced role-specific, customized with company data/knowledge, security certification.	None	Security awareness training is performed.	Security on-demand training and advanced-role specific training are performed. Security centralized reporting knowledge is used.	Material specific to company history is used in training. Vendors and outwourced workers are trained. Annual training required for everyone.	Progression on security training curriculum is rewarded.
Publish Operations Guide	Guiding coverage	Basic (critical security information for deployment), moderate (procedures for typical application alerts), thorough (formal operational security guides).	None	Regular operations guide with critical security instructions for deployment.	Moderate operations guide with critical security instructions and procedures for typical application alerts.	Thorough operations guide with with detailed security instructions and, procedures for all application alerts.	Very thorough operations guide with with maximal security instructions and, procedures for all application alerts.

SECU Rating Scale

Practices Consolidation → Practices Grouping → Practices Summarization



Task	Practices	Characteristics for SECU ratings	Low	Nominal	High	Very High	Extra High
Requirements and Design	Apply Security Requirements	Requirements specification	Ad-hoc security requirements.	Basic security requirements derived from business functionality.	Moderate security requirements derived from business functionality and compliance drivers.	Complex security requirements derived from business functionality, compliance drivers and known risks.	Extreme security requirements derived from business functionality, compliance drivers and application/domain specific security risks.
	Security Features	Scope and rigour	None.	Build basic security features (authentication, role management, key management, audit/log, cryptography, protocols).	Build additional security features (authentication, role management, key management, audit/log, cryptography, protocols).	Build secure-by-design middleware for security features (authentication, role management, key management, audit/log, cryptography, protocols).	Build container-based approaches for security features (authentication, role management, key management, audit/log, cryptography, protocols).
	Apply Threat Modeling	Attack information	None.	Ad-hoc threat modeling.	Apply threat modeling with generic attacker profiles.	Apply threat modeling with specific attackers information.	Apply threat modeling using new attack methods developed with a science team.
Coding	Apply Secure Coding Standards	Standards coverage	Ad-hoc secure coding	Address common vulnerabilities	Address common and off-nominal vulnerabilities	Address all vulnerabilities and some weakness	Coding to address all known vulnerabilities and weaknesses
	Apply Security Tooling	Tools usage	Casual use of standard static analysis tool to identify security defects.	Basic use of static analysis tool to identify security defects.	Routine use of static analysis and penetration testing tools to identify security defects.	Extensive use of static analysis, penetration testing and black-box security testing tools.	Rigorous use of static analysis, penetration testing and black-box security testing tools with tailored rules.
Verification and Validation	Perform Security Testing	Testing rigour and coverage	Ad-hoc security testing	Basic adversarial testing	Moderate adversarial testing driven with security requirements and security features.	Extensive adversarial testing driven by high security risks.	Rigorous adversarial testing driven by security risks and attack patterns.
	Perform Security Review	Review rigour and coverage	Ad-hoc security features code review.	Basic security features code review.	Moderate security code review.	Systematic extensive security code and design review.	Systematic rigorous security code and design review.
	Perform Penetration Testing	Penetration testing frequency	Ad-hoc penetration testing.	Basic penetration testing addressing common vulnerabilities (for sanity check before shipping).	Routine penetration testing (each release) addressing common and critical vulnerabilities.	Frequent penetration testing (each increment) based on project artifacts.	Deep-dive analysis and maximal penetration testing.

Application Type	Stand alone	Network Connected	Access to private data	Hospitals, Banking	Loss of life
General Characteristic	None/Ad-hoc	Basic	Moderate	Extensive	Rigorous
Requirements and Design Summary	Security requirements considered part of reliability.	Basic security requirements and features. Basic threat modeling.	Moderate security requirements and additional security features (audit/log, cryptography). Moderate threat modeling	Complex security requirements, advanced secure-by-design security features middleware development. Threat modeling with specific attackers information.	Extreme security requirements, container-based approaches for advanced security features development. Rigorous threat modeling.
Effort Multiplier	1	1.1	1.25	1.45	1.7
Coding and Tools Summary	No secure coding and no use of static analysis tool.	Basic vulnerabilities applicable to the software will be prevented with secure coding standards or detected through basic use of static analysis tools.	Known and critical vulnerabilities applicable to the software will be prevented with secure coding standards or detected through routine use of static analysis and penetration testing tool.	Extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards or detected through extensive use of static analysis, black-box and penetration testing tools.	Very extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards or detected through rigorous use of static analysis, penetration testing and black-box security testing tools with tailored rules.
Effort Multiplier	1	1.1	1.2	1.35	1.5
Verification and Validation Summary	No security testing, review or penetration testing.	Basic adversarial testing and security code review. Basic penetration testing before shipping.	Moderate adversarial testing and security code review. Routine penetration testing each release.	Extensive adversarial testing and security design/code review. Frequent and specialized penetration testing each increment.	Rigorous adversarial testing and security design/code review. Exhaustive deep-dive analysis penetration testing.
Effort Multiplier	1	1.2	1.4	1.65	2
Security Activities One-Line Summary	Security-related activities for requirements, coding, and testing considered part of reliability.	Basic security-related activities for requirements, coding, and testing. Typical security functional features. Regular use of static analysis tools to detect security defects.	Moderate security-related activities for requirements, design, coding, and testing. Additional security features (audit/log, cryptography). Identification and controlled update of third-part components' security patches. Routine use of static analysis and penetration testing tools.	Complex security requirements and threat modeling. Advanced secure-by-design security features. Extensive adversarial testing and security design/code review. Security assessment of third-part components and timely security patches updates. Thorough use of statics analysis, black-box, and penetration testing tools.	Extreme security requirements and threat modeling. Container-based approaches to advanced security features. Exhaustive adversarial testing, security design/code review, and deep-dive analysis penetration testing. Third-part components rigorously assessed and updated by a security science team. Maximal use of tools for static analysis, penetration testing, and black-box security testing.

Summarized Practices

General Characteristic	Basic	Moderate	Extensive	Rigorous
Characteristics	High	Very High	Extra High	Ultra High
Security Requirements and Design Summary	Basic security requirements and features. Basic threat modeling.	Moderate security requirements and additional security features (audit/log, cryptography). Moderate threat modeling.	Complex security requirements, advanced secure-by-design security features middleware development. Threat modeling with specific attackers' information.	Extreme security requirements, container-based approaches for advanced security features development. Rigorous threat modeling.
Secure Coding and Security Tools Summary	Basic vulnerabilities applicable to the software will be prevented with secure coding standards and/or detected through basic use of static analysis tools.	Known and critical vulnerabilities applicable to the software will be prevented with secure coding standards and/or detected through routine use of static analysis tools.	Extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards and/or detected through extensive use of static analysis and black-box tools.	Very extensive list of vulnerabilities and weaknesses applicable to the software will be prevented with secure coding standards and/or detected through rigorous use of static analysis and black-box security testing tools with tailored rules. Employ formal methods in coding.
Security Verification and Validation Summary	Basic adversarial testing and security code review. Basic penetration testing. Security V&V activities conducted within the project.	Moderate adversarial testing and security code review. Routine penetration testing. Security V&V activities conducted by an independent group.	Extensive adversarial testing and security design/code review. Frequent and specialized penetration testing. Security V&V activities conducted by an independent group at the organizational level.	Rigorous adversarial testing and security design/code review. Exhaustive deep-dive analysis penetration testing. Use of formal verification and custom developed V&V tools. Security V&V activities conducted by an outside certified company.

One-Line Summary

None/Ad-hoc	Basic	Moderate	Extensive	Rigorous
Nominal	**High**	**Very High**	**Extra High**	**Ultra High**
Security-related activities for requirements, coding, and testing nonexistent.	Basic security-related activities for requirements, coding, and testing. Typical security functional features. Regular use of static analysis tools to detect security defects within the project.	Moderate security-related activities for requirements, design, coding, and testing. Additional security features (audit/log, cryptography). Identification and controlled update of third-part components' security patches. Routine use of static analysis and penetration testing tools. Security V&V activities conducted by an independent group.	Complex security requirements and threat modeling. Advanced secure-by-design security features. Extensive adversarial testing and security design/code review. Security assessment of third-part components and timely security patches updates. Thorough use of statics analysis, black-box, and penetration testing tools. V&V activities conducted by an independent group at the organization level.	Extreme security requirements and threat modeling. Container-based approaches to advanced security features. Exhaustive adversarial testing, security design/code review, deep-dive analysis penetration testing, and use of formal methods throughout the lifecycle. Third-party components rigorously assessed and updated by a security science team. Maximal use of tools for static analysis, penetration testing, and black-box security testing. Use of formal verification and custom developed V&V tools. Security V&V activities conducted by an outside certified company.

Outline

- Background
- Previous studies on secure software development costs
- Proposed rating scale
- ➢Data collection and Initial results of a Wide-band Delphi
- Next steps

Data Collection

Security experts estimates for the security parameter

Estimation experts estimates for the security parameter

} Wideband Delphi

Industry — Projects' Data → Manual Data Collection Form

OSS — Projects' Data → Automated Data Collection

\+ 

Projects' Data → Survey

Delphi Results

- Productivity Range (PR)
  - PR is the degree of influence a model parameter has on the estimated effort.
  - The Delphi session asked participants to rate the PR for security
  - The result is an average of 2.17 (table below)
  - A range of 2.17 means the parameter ratings from Nominal to Ultra High can change the effort estimate up to 117%, i.e., it costs 117% more to implement extreme security requirements and threat modeling

Security Parameter Characteristic	Average (AVG)	Standard Deviation (SD)	Coefficient of Variation (SD/AVG)
Security Requirements and Design	1.18	0.045	4%
Security Coding and Tools	1.18	0.130	11%
Security Verification & Validation	1.56	0.152	10%
Total Productivity Range	2.17		

Converting Delphi to a Rating Scale

- The Productivity Range is used to derive individual quantitative values, EM_i, for each rating level based on exponential growth
- Exponential growth is used in all COCOMO III parameters

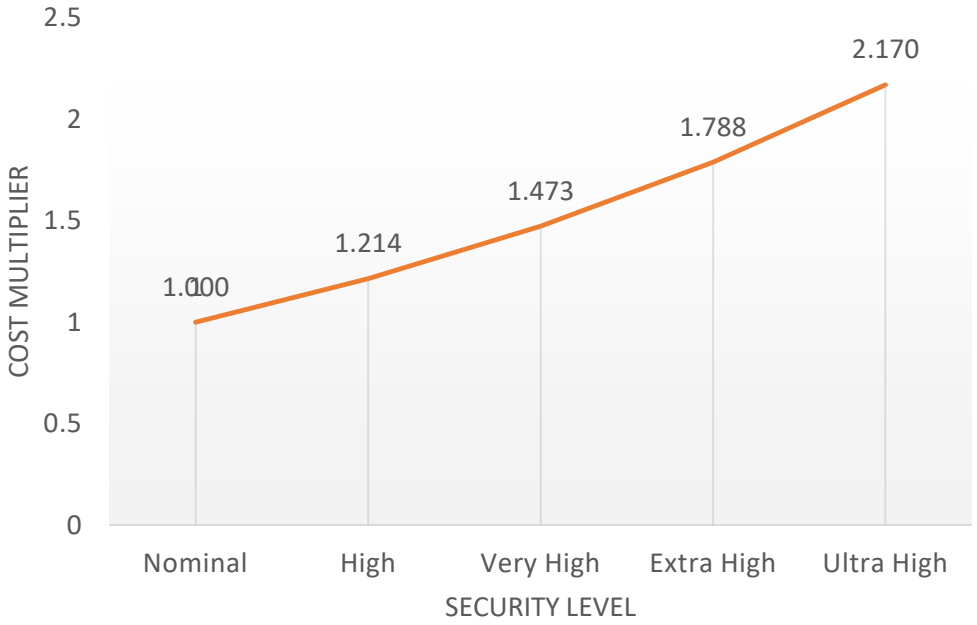
Exponential Growth

$$EM_i = (1 + r)^i$$

$$r = range^{(1/n)} - 1$$

- EM_i is the effort multiplier for level *i*
- *i* is the security level (Nominal is 0)
- *r* is the growth rate
- *range* is the range of the effort multiplier
- *n* is the index of the highest level *(i=4)*

Security Rating Scale



Outline

- Background
- Previous studies on secure software development costs
- Proposed rating scale
- Data collection and Initial results of a Wide-band Delphi
- ➢Next steps

Next Steps

- Collect project and OSS data
- Calibrate the COCOMO III model
  - Functional size measures as well as traditional lines of code
  - Domain classification of software applications
  - All the model parameters
- Validate the security rating scale and its impact on effort

Thank you!

Elaine Venson
venson@usc.edu
Brad Clark
clarkbk@usc.edu

September 16, 2020

References

- Chad Heitzenrater, Rainer Bohme, Andrew Simpson, 2016. The Days Before Zero Day: Investment Models for Secure Software Engineering, in: Proceedings of the 15th Workshop on the Economics of Information Security (WEIS). Presented at the Workshop on the Economics of Information Security.
- McGraw, G., 2006. Software Security: Building Security In, 1 edition. ed. Addison-Wesley Professional, Upper Saddle River, NJ.
- OWASP SAMM Project, 2017. Software Assurance Maturity Model (SAMM): A guide to building security into software development - v1.5 (No. Version 1.5).
- Rashid, A., Chivers, H., Danezis, G., Lupu, E., Martin, A., 2019. The Cyber Security Body of Knowledge.
- Sammy Migues, John Steven, Mike Ware, 2019. Building Security in Maturity Model (BSIMM) - Version 10 (No. 10). Synopsys Software Integrity Group.
- Tierney, J., Boswell, T., 2017. Common Criteria: Origins and Overview, in: Mayes, K., Markantonakis, K. (Eds.), Smart Cards, Tokens, Security and Applications. Springer International Publishing, Cham, pp. 193–216. https://doi.org/10.1007/978-3-319-50500-8_8
- Venson, E., Guo, X., Yan, Z., Boehm, B., 2019. Costing Secure Software Development: A Systematic Mapping Study, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES '19. ACM, New York, NY, USA, pp. 9:1–9:11. https://doi.org/10.1145/3339252.3339263
- Venson, E., Alfayez, R., Marília M. F., G., Rejane M. C., F., Boehm, B., 2019. The Impact of Software Security Practices on Development Effort: An Initial Survey, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). Presented at the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–12. https://doi.org/10.1109/ESEM.2019.8870153