Driving Secure Software Initiatives Using FISMA: Issues and Opportunities

Robin Gandhi, University of Nebraska at Omaha
Keesha Crosby, Tri-Guard Risk Solutions, LTD
Harvey Siy, University of Nebraska at Omaha
Sayonnha Mandal, University of Nebraska at Omaha

Abstract. Federal agencies install many security controls for Federal Information Security Management Act (FISMA) implementation. National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53 revision 4 (rev4) standardizes these security and privacy controls. This article presents a study of NIST SP 800-53 security controls. The purpose is to classify the security controls from dimensions relevant to software security. This classification highlights issues and motivates opportunities to drive software security initiatives using FISMA.

Introduction

FISMA mandated security controls drive many information security programs in the federal government. But their impact on the development and/or acquisition of secure software is not well understood. Secure software (or software assurance) provides the basis for the belief that it will operate as expected in its threat environment. Such software has capabilities to resist most attacks. It can tolerate as many as possible of those attacks it cannot resist. Finally, it can contain the damage and recover to a normal level of operation as soon as possible. This article outlines a method to classify security controls based on dimensions relevant to secure software. The findings highlight issues and motivate opportunities for driving secure software initiatives using FISMA.

Let's begin by taking a look at the source of FISMA mandated security controls for a federal information system. As part of the FISMA implementation project [1], NIST has produced several key security standards and guidelines. This includes the Federal Information Processing Standard (FIPS) 199, FIPS 200, and NIST SP 800-53. Guidance within these documents work hand-in-hand for executing the first two steps in the NIST Risk Management Framework (NIST SP 800-37). Step 1 requires security categorization of information and information systems. The potential for impact to confidentiality, integrity, and availability of information determines the categorization. FIPS 199 establishes standards for categorizing information systems in this step. Step 2 requires selection of security controls based on the security categorization in step 1. In this step FIPS 200 establishes the low, moderate and high security baselines for control selection. A baseline is a set of minimum security controls defined for a low-impact, moderate-impact, or high-impact information system [2]. NIST SP 800-53 documents these control baselines as part of a larger control catalog [2].

NIST SP 800-53 specifies controls at the level of an organization or information system. There are many mandatory controls for perimeter security, system integration, operations, and organizational processes. But controls for building-security-in the information system components, i.e. software, are hard to discern. These controls are often tangled with other system concerns. In particular, organization or system level controls need expert interpretation for secure software relevance. This article addresses this issue by the development of a coding instrument. The instrument inquires the relevance of a control along the many dimensions of secure software.

The paper is organized as follows. Section 2 outlines the development of a coding instrument. Section 3 enumerates findings from applying the instrument to NIST SP 800-53. The findings provide insights that were not accessible before due to the large volume of the control catalog. They highlight software assurance practices buried within a larger organizational and information system context. To conclude, section 4 summarizes issues and opportunities identified from this study.

Coding Instrument for Software Assurance

Bootstrapping development of a coding instrument requires a recognized definition of software assurance. NIST SP 800-53 states the definition of assurance from a system perspective. Its focus is on the emergent behavior of the components for meeting the security requirements of the system. A narrower focus on software assurance exists in many definitions by government agencies (e.g. NASA, CNSS, DHS, etc.), focus groups (e.g. SAFECode) and academics/researchers. Finally, the following definition became basis of the coding instrument. CERT/SEI has also adopted this definition for their Masters in Software Assurance curriculum project.

"Software Assurance is the application of technologies and processes to achieve a required level of confidence that software systems and services function in the intended manner, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures [3]."

Further analysis identified more dimensions. These include dimensions related to developers, software artifacts, policies, operations, weaknesses and lifecycle processes. These provide a more holistic perspective of software assurance within the coding instrument. A brief summary of the resulting 18 coding dimensions in the instrument is as follows:

The control …

… requires the application of process for software assurance: (P)

… requires the application of technology for software assurance: (T)

… requires the application of process and technology combined for
software assurance: (P+T)

…is not directly applicable to software assurance: (N)

…is withdrawn from the control catalog: (W)

The control is relevant to …

… a developer involved in the software construction or maintenance: (Developer)

… the implementation of software artifacts: (Artifact)

… software lifecycle processes: (Lifecycle)

… policies to be enforced by software: (Policy)

… software weaknesses to be avoided, accidental or intentional vulnerabilities in software or its use, or if compliance assessment with the control will fail due to a software weakness: (Weakness/Vulnerability/Failure)

… processes for software use, configuration or maintenance: (Operations)

… security capabilities to be provided by software in a threat environment: (Capability)

… software recovery from intrusions and failures: (Recovery)

The wording of NIST SP 800-53 control descriptions have direct implications on enforcement. Control refinements by interpretation in the context of software can impact enforceability. Thus, the instrument distinguishes control descriptions that are enforceable for software security. Controls which need refinement to apply in the context of software are also identified. The instrument accomplishes these tasks as follows:

The control description …

… explicitly constraints information system components, software, code, services or applications: (E)

… implicitly constrains software components, requiring expert interpretation: (I)

… as part of the supplemental guidance refers to software components, but not in the regulatory-enforced description: (IS). This considered a subset of (I).

Each security control beings with the phrase “the organization” or “the information system.” These terms are defined as: “… information system refers to those functions that generally involve the implementation of information technology (e.g., hardware, software, and firmware). Conversely, the term organization refers to activities that are generally process- driven or entity-driven—that is, the security control is generally implemented through human or procedural-based actions. Security controls that use the term organization may still require some degree of automation to be fulfilled.” [2] This characteristic of a control description is captured as follows:

The control description…

… starts with the phrase “The organization”: (ORG)

… starts with the phrase “The information system” (SYS)

The authors applied the final instrument to investigate each NIST SP 800-53 security control. This includes controls in 26 families, including the new privacy families. A total of 958 security controls, including control enhancements, were part of the study.

To begin the study, the four authors of this article reviewed each control. Later in a group session the authors discussed controls with divergent categorizations. The authors performed peer evaluations of early coding efforts to ensure consistent instrument use. These peer evaluations helped identify and remove sources of ambiguity early in the process.

The process resulted in a preliminary list of software assurance related controls. For feedback the authors disseminated these controls using the NIST software assurance mailing list. Several community members provided feedback. Based on the feedback and internal team review, the authors added 17 controls to the initial set of 535 controls. This brought the total number of software assurance related controls to 552. This list of security controls can is available here [4].

Observations and Findings

This section reports observations from applying the instrument to NIST SP 800-53 security controls. A brief discussion of significance follows each observation.

Do NIST SP 800-53 controls emphasize software assurance related topics?

- Target of observation: # of controls and control families relevant for software assurance
- Observed data:
    - 57.62% (552/958) controls are relevant to software assurance
    - 69.23% (18/26) families have controls relevant to software assurance
- Significance: Software is a key element in a majority of information system components. The instrument observed relevance for software assurance across NIST SP 800-53 security controls.

How obvious are the control interpretations for software security?

- Target of observation: # of Explicit and Implicit controls
- Observed data:
    - 189 controls are Explicit
    - 363 controls are Implicit
- Significance: A large number of implicit controls show a significant burden for stakeholders in the A&A activities. Stakeholders need to interpret, negotiate and agree upon implicit software assurance related controls.

How many controls related to software security are strictly enforceable?

- Target of observation: # of Explicit controls assigned to baselines
- Observed data:
    - 16 explicit controls are assigned to the LOW baseline.
    - 38 explicit controls are assigned the MODERATE baseline.
    - 53 explicit controls are assigned the HIGH baseline.
    - 342 controls (62%) are not assigned to any baseline.
- Significance: A small percentage of software assurance related controls are enforceable by minimum security baselines. Furthermore, a large number of software assurance controls remain unassigned to any baseline. The A&A activities rely on the effectiveness of the tailoring step to select the unassigned controls. Tailoring activities adjust the control baselines to a level commensurate with the perceived risk.

Which control families have a high density of explicit software assurance controls?

- Target of observation: % of software assurance controls with the Explicit (E) coding dimension within a control family
- Observed data:
    - Percentage of software assurance controls with the

Explicit (E) coding dimension within control families. See figure 1.

- Significance: Figure 1 shows that explicit controls are concentrated in the SA, SI, SC and CM families. This observation aligns well will the assertions made by NIST SP 800-53 authors. They have described SA, SI and SC control families with the most emphasis on software assurance. There is one surprising observation. The AC family has the least amount of explicit controls. But this family also has the most number of software assurance controls (95 software assurance controls).

What topics do explicit software assurance controls focus on?

- Target of observation: Co-occurrence of Developer, Artifact, Lifecycle, Policy, Weakness/Vulnerability/Failure, Operations, Capability and Recovery coding dimensions with the Explicit coding dimension.
- Observed data:
  - 61.38% Operations
  - 47.09% Capability
  - 39.68% Artifact
  - 36.51% Developer
  - 32.80% Lifecycle
  - 20.11% Weak/Vuln/Fail
  - 15.87% Policy
  - 7.94% Recovery
- Significance: The observed data shows that explicit control descriptions are not balanced. NIST SP 800-53 authors bias them more towards software use, configuration, maintenance and functional capabilities. Guidelines for developers, software artifacts and lifecycle activities form the next set of biases. Finally, weakness/vulnerability/failure topics seem to get much less mention compared to other topics.

Which control families have the most number of software assurance controls? What software assurance topics do they cover?

- Target of observation:
  - Within a control family
    - # of software assurance related controls within a family (includes both explicit and implicit controls)
  - frequency of occurrence for Developer, Artifact, Lifecycle, Policy, Weakness/Vulnerability/Failure, Operations, Capability and Recovery coding dimensions within a family as well as across families
- Observed data:
  - # of software assurance related controls within a family as shown in Figure 2
- Frequency of occurrence for Developer, Artifact, Lifecycle, Policy, Weakness/Vulnerability/Failure, Operations, Capability and Recovery coding dimensions in a control family. Top three coding dimensions in the top five control families with the most number of software assurance controls are shown in Figure 3.
- Significance: The most number of software assurance related controls come from the AC family. This family also has a high percentage of implicit controls. Like this family, other control families also exhibit tangled and hidden software assurance concerns.

The next observations show that Operations and Capability dimensions dominate several control families. But, the other co-occurring dimensions do reflect the focus of the family. For example, Artifact, Developer and Lifecycle dimensions best characterize the SA family. Policy dimension best characterizes AC and IA family. Finally, the Weakness/Vulnerability/Failure dimension best characterizes SI family.

The Developer dimension frequently overlaps with Artifacts, Lifecycle, Operations dimensions. This is because process-related activities (Lifecycle and Operations) involve developers. They also produce software artifacts that have to meet certain standards.

Next, there are many more controls coded as Operations (337) than Lifecycle (113). This implies that NIST SP 800-53 controls focus more on operational issues than on the development process. Roughly half of the controls coded as Lifecycle are also coded as Operations (51/113). This implies that many of the lifecycle processes are also biased towards operations.

Finally, there are many more controls coded as Capability (319) than Artifact (151). This implies that NIST SP 800-53 controls emphasize functional security requirements a lot. But focus less on placing requirements on the artifact creation process.

How do "organization" related controls compare to "information system" related controls for software assurance?

- Target of observation: Co-occurrence of ORG and SYS

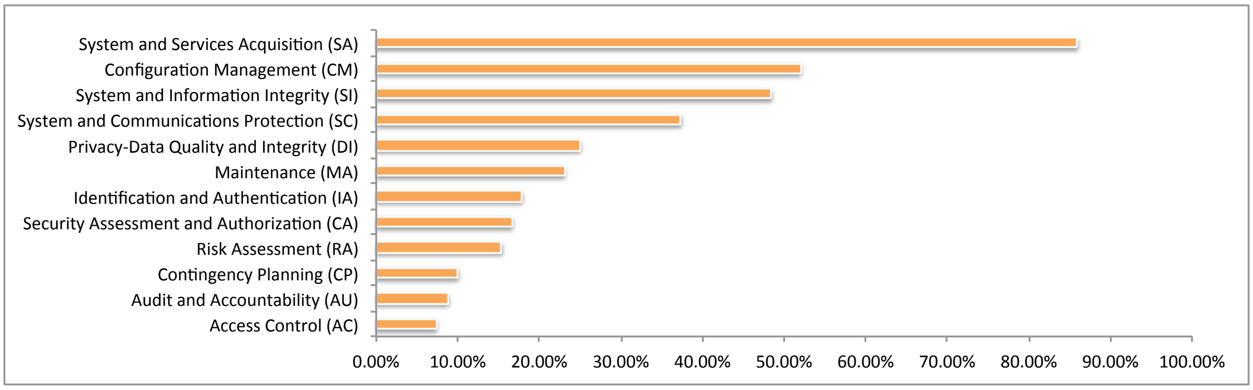


Figure 1: Percentage of software assurance controls with the Explicit (E) coding dimension within control families

Figure 2: Total # of software assurance controls across all families

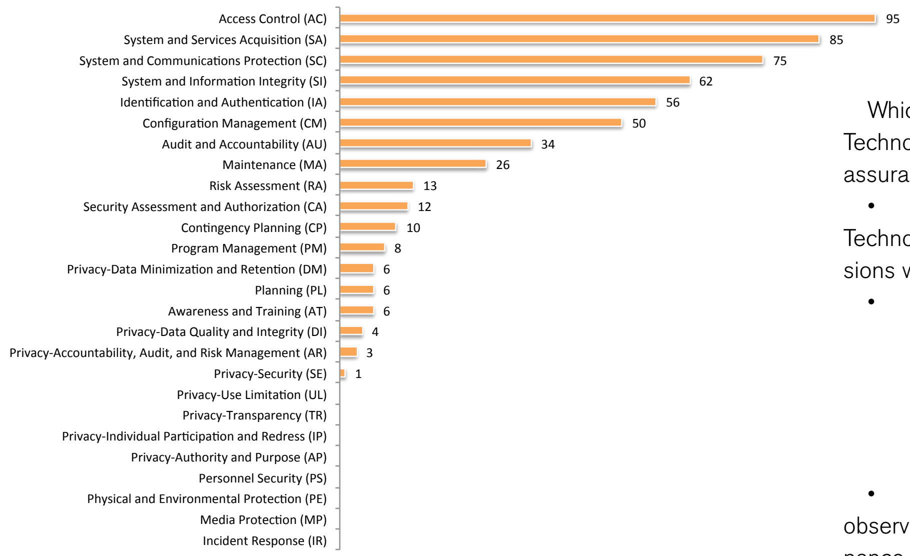
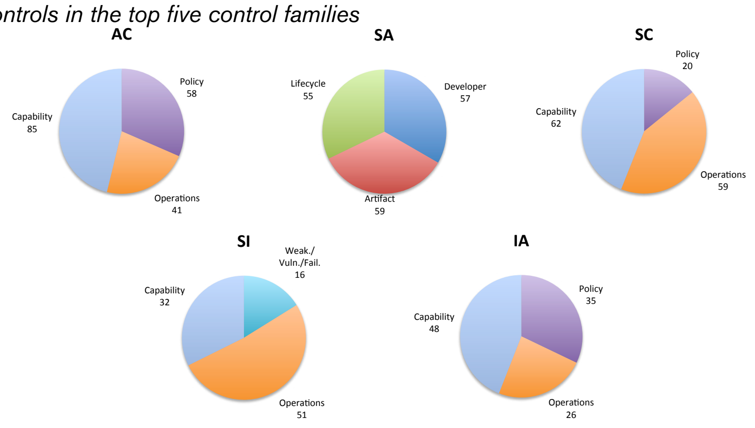


Figure 3: Distribution of Developer, Artifact, Lifecycle, Policy, Weakness/Vulnerability/Failure, Operations, Capability and Recovery coding dimensions for controls in the top five control families



coding dimensions with respect to Process (P), Technology (T), (P+T), and Explicit (E) coding dimensions

- Observed data:
  - 93.62% (191/204) of SYS controls are purely Technology-oriented
  - 99.7% (347/348) of ORG controls are Process-oriented,
  - 51.14% (178/348) of ORG controls have both a Process and Technology component (P+T).
  - ~45% (154/348) of ORG controls are Explicit
  - ~17% (35/204) of SYS controls are Explicit.
- Significance: This observation confirms that SYS controls suggest technological solutions. Next, almost all ORG controls are process oriented. But more than half of these controls also have a technological component. These observations align well with the definition of SYS and ORG in NIST SP 800-53.

Only a small percentage of explicit controls are SYS compared to ORG. This suggests that explicit controls recommend process and technology combined solutions over just technology.

Which control families predominantly emphasize Process, Technology or Process and Technology combined for software assurance?

- Target of observation: % of controls with Process (P), Technology (T) or Process and Technology (P+T) coding dimensions within a control family
- Observed data:
  - Over 50% Process (P): MA (19/26), SA (55/85)
  - Over 50% Technology (T): AC (67/95), AU (20/34), IA (29/56), SC (41/75)
  - Over 50% Process and Technology (P+T): CA (10/12), CM (28/50)
- Significance: The following justifications explain these observations. Process based controls are common for maintenance and acquisition activities. Emphasis on process in MA and SA control families reflects this. Next, automatic control mechanisms are common for access control, audit, identification, authentication and secure communications. Emphasis on technology in AC, AU, IA and SC control families reflects this. Finally, automated mechanisms often support manual processes of performing security assessments. Emphasis on process and technology combined in CA and CM families reflects this.

Issues and Opportunities

Information systems are software intensive. As a result, weaknesses in software components present a significant source of risk. Due to many implicit software assurance controls these risks are not well understood in the context of FISMA. This makes it difficult to manage software assurance as a first-class entity in the system lifecycle. Furthermore, in new system acquisitions, stakeholders tailor security controls based on system needs. During tailoring, stakeholders address many system security controls. But software security gets less attention than it deserves. Thus, the system matures with unmitigated software deficiencies and flaws. At the same time, software evolves with new features and capabilities more rapidly than the system. This fact is also evident during security test and evaluation as well as operational test and evaluation. During these assessments software components are many development versions ahead of the system maturity.

The Common Weakness Enumeration (CWE) provides a unified and measurable set of weaknesses. But, this large enumeration of over 700 weaknesses presents a significant cognitive challenge. A&A stakeholders need to understand what weaknesses are most relevant to security controls. Yet many security controls do not provide any CWE selection guidance. For example, supplemental guidance for the SI-2 Flaw Remediation control states the following. "...Organizations take advantage of available resources such as the Common Weakness Enumeration (CWE) or Common Vulnerabilities and Exposures (CVE) databases in remediating flaws discovered in organizational information systems." It is encouraging to see the mention of standard enumerations of software weakness like CWE. But, assessing compliance with this control will likely not be repeatable or uniform.

Finally, controls to avoid some of the most egregious software weaknesses do exist in the catalog. For example, controls for static code analysis, threat and vulnerability analyses exist. But they are not assigned to any control baseline (not even high-impact!). NIST SP 800-53 categorizes them as assurance controls. Stakeholders can use these as needed in different situations but not mandated. So it becomes easy for a software developer or organization to just "tailor these controls out." Tools that support security A&A activities make it even easier to filter out these controls.

While there are issues, as with any other A&A process, many opportunities also exist. NIST SP 800-53 rev 4 control catalog has a comprehensive set of requirements to develop or procure secure software. But the FIPS 200 minimum security baselines needs to include them in the low, moderate, or high-impact baselines. To build-security-in, the bar needs to be raised.

NIST SP 800-53 controls are often specified independent of specific technologies and platforms. As a result, the controls align well with abstract "Class" and "Base" level software weaknesses in the CWE. Thus, developing mappings between security controls and standard software weaknesses is essential. This effort is currently being undertaken using assurance cases as a mapping mechanism. These results are beyond the scope of this article. Finally, many relationships exist among controls as well as among CWEs. These will be essential to unravel the cascading dependencies among system components.

Just regulatory processes and controls alone cannot guarantee secure software. But, they do play a significant role in making software security programs a strategic priority. Our goal is to make software assurance related controls easily understood, communicable, and manageable. These attributes are an essential precursor to measure control effectiveness for software security. That is if the controls do in fact lead to reduction in security weaknesses in software. Also, prove the impact of these controls for acquiring software that can be securely configured, deployed and maintained.

Acknowledgements

This research is partially funded by the US Department of Homeland Security Science and Technology Directorate Cyber Security Division, D13PCC00228, "An Engagement to Look Forward to: Security Requirements and Software Weaknesses." We also thank the anonymous reviewers for their thoughtful comments on previous drafts of this paper.

ABOUT THE AUTHORS

Robin A. Gandhi, Ph.D. is an Associate Professor in the College of Information Science and Technology at the University of Nebraska, Omaha. He received his Ph.D. from The University of North Carolina at Charlotte. The goal of Dr. Gandhi's research is to develop theories and tools for designing dependable software systems that address both quality and assurance needs. He is a member of DHS Software and Supply Chain Assurance Working Group on Workforce Training and Education.

Keesha M. Crosby is the Founder and CEO of Tri-Guard Risk Solutions, Ltd (T-GRS). Prior to founding T-GRS, Crosby served as industry and government subject matter expert in software assurance as well as system security engineering arena. She has authored articles for IEEE and several patents pending. T-GRS has a tool called SACRE (Software Assurance Compliance verification and Risk Evaluation) which automates the decision making for developers and auditors based on weaknesses likelihood of breach.

Harvey Siy, Ph.D., is an associate professor in the Department of Computer Science at the UNO. He received his doctorate in computer science from the University of Maryland at College Park. He conducts empirical research in software engineering to understand and improve technologies that support the development and evolution of reliable software-intensive systems. Siy has previously held positions at Lucent Technologies and its research division, Bell Laboratories.

Sayonnha Mandal is currently pursuing her Ph.D. in Information Technology at the University of Nebraska, Omaha.

REFERENCES

1. "FISMA Implementation Project." NIST Computer Security Division. N.p., n.d. Web. 28 May 2014. <http://csrc.nist.gov/groups/SMA/fisma/index.html>.
2. United States. National Institute of Standards and Technology. SP 800-53 Revision 4. Security and Privacy Controls for Federal Information Systems and Organizations. N.p., Apr. 2013. Web. <http://csrc.nist.gov/publications/PubsSPs.html#800-53>.
3. Mead, N., Allen, J., Ardis, M., Hilburn, T., Kornecki, A., Linger, R., & McDonald, J. (2010). Software Assurance Curriculum Project Volume I: Master of Software Assurance Reference Curriculum (CMU/SEI-2010-TR-005). Retrieved November 07, 2013, from the Software Engineering Institute, Carnegie Mellon University website: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9415>
4. Draft list of software assurance related NIST SP 800-53 rev4 controls: <http://faculty.ist.unomaha.edu/rgandhi/swa/controls.pdf> <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9415>