

Comments Abstract / Executive Summary:

NIST.SP.800-218r1.ipd.pdf

About Mike Sanderson:

I am an application software developer and since 2013 have worked on iPhone apps, working on projects with high security requirements, including as a vendor for a launch partner for Apple Wallet NFC in 2015, on a secure messaging app in widespread use in high-security settings, handling user health data, and independent security research. I am based in New York City and have delivered multiple presentations to professional development groups since 2016. My comments come from the software development perspective, including extensive experience with API design and use.

See attached Excel file for line-referenced comments.

Executive summary / narrative comment:

The framework is welcome, and puts an important focus on the need for secure software development practices.

These comments are primarily concerned that the premise of the SSDF as a “high-level” (line 169) overview with a focus “on the outcomes of the practices rather than on the tools” ((line 164-165) is likely to be frustrating to working professionals at all levels of software development. No one working in software would not implement these practices if they could wave a magic wand and implement them. The issue is who, why, and how.

Therefore, large portions of this SSDF are, in fact, undergirded by information-gathering, training, and education. Everyone wants to do these things, and can’t until they learn how. While the “references” column is the right direction, in the spot-checks, these single-references tend to point to very specific items in the referenced documents, and create a gap between the SSDF practice and the specific technique.

In short, most of these items are in fact, having a preparatory step: *undertake information-gathering using sector-specific resources and platform-specific resources.*

While that may be obvious, the tone does not reflect that this is in fact the hard part. Instead, the document seems premised on the idea that individuals who consult it are looking for guidance to be told to do these things, and not who, how and why to do them.

Who, Why, and How.

Who:

Some of the practices should be implemented by the operating system, the tool vendor, the open-source project in use, a community group, or the organization. Within an organization, it is obvious to the point of tautology that different individuals should be responsible for roles.

References to industry-standard terms like “lead developer”, “junior developer”, “project lead”, “technical lead”, etc., will help organizations. The role of forming requirements would be helpful to link to tasks like P.O. 1.1 (about identifying external requirements) and P.O. 4.4 (incorporating into agile workflows.) While many software companies and organizations are organized differently, there are some commonalities that would it would be helpful for this framework to engage with.

Why:

In the “Notional Implementation Examples,” there is no information about *why* an organization would undertake this example. Why characteristics of the organization or task would lead to the decision to implement this example. Many of the examples are quite high. Without the reason why, these examples actually make it harder to think of not-listed examples for other cases.

How:

This document does not proclaim how, but no information is given — other than referencing to very specific items — as to how. What is needed is a systematic understanding of on what level the implementation of the practices should be developed. This is especially concerning because many of these practices do not need to be implemented on a per-organization level. Instead, the platform owner, the development toolmakers, or the underlying frameworks should find common solutions. This intermediate level — known as “community profiles” in the NIST Cybersecurity Framework 2.0 — is not what this document is supposed to do, but this document would do much better to acknowledge that this level exists.

#	Organization	Commentor	Comment's Type (General [G], Technical [T], Editorial [E])	Line #	Section	Comment(Include rationale for comment and proposed change)	Suggested change
1	(Independent)	Mike Sanderson	E	155-163	Executive Summary	The use of "should" in these four bullet points is frustrating, because no one in software needs to be told they "should" do these things. People don't know how to do them. Yet this and the subsequent areas do not even allude that it is aware that most software developers or organizations have no idea how to get the resources on how to do these things — or how to evaluate the resources that are out there.	Acknowledge that investigating how to do these things on a per-sector, per-platform basis, and evaluating authoritative guidance, will be much of the work of doing them.
2	(Independent)	Mike Sanderson	E	164-169	Executive Summary	Saying " The SSDF does not prescribe how to implement each practice" and that it can be "used" by anyone is to say it doesn't contain the information that the users actually need.	Make reference to "Community profiles" which are trying to bring attention to the possibility of shared authoritative sources.
	(Independent)	Mike Sanderson	E	202-206)	1. Introduction	Again, this is the concept of "community profiles" which is part of the solution. How to identify these community profiles and to tailor them is the whole problem.	Make reference to "Community profiles"
	(Independent)	Mike Sanderson	E		1. Introduction	"This is similar to the approach taken by the Framework" ... concerning that one framework is "similar" to another. Shouldn't the SSDF be within NESTS? That is, someone is implementing NISTCSF, and within that, implementing NISTSSDF?	Change "similar to" to "within"
	(Independent)	Mike Sanderson	E		1. Introduction	The reference to "Business owners" and "project managers" is so important, because they must be involved. Developers typically are fully accountable for their time. Every software project is ultimately run by a project manager accountable to the business side, because decisions large and small often, immediately or ultimately, reduce to trade offs about use of resources, and the person accountable for the use of resources must understand the trade-off and make the call.	Note that business owners and project managers should be aware that resources may not be available, and that investigating may take time. Furthermore, they should be aware of the tradeoffs in time from demanding protections that aren't needed.
	(Independent)	Mike Sanderson	E	277-278	1. Introduction	"tenant organization"? I have never heard this phrase and it is barely clear in the context it is used. It is distracting.	Replace tenant organization
	(Independent)	Mike Sanderson	E	290-291	2. Secure Software Development Framework	"Protect the Software " The use of "software" here is extremely confusing, but the point is crucial: Orgs must protect the software they use to protect the software they develop. Without a secure development environment, secure software development is impossible.	The term that would make the most sense to the software developers is probably "development environment" or "software tooling" or "software tools". For some reason, "tooling" is widely used. Also maybe specify "development and deployment software".
	(Independent)	Mike Sanderson	E	314–316	2. Secure Software Development Framework	The table uses terms like "sensitive data," "qualified person," and "well-secured," which are not defined in this publication. Organizations that adopt the SSDF should define these terms in the context of their own environments and use cases." These are not issues of definition. These terms have definitions. They are terms of risk assessment. Again, these high-level terms are "what" should be done, and organizations must analyze "why" to answer "how".	Not definition but "identify with risk assessment"
	(Independent)	Mike Sanderson	E	322	Table 1. Secure Software Development Framework (SSDF) Version 1.2	This is ridiculous to have as one line number for the entire framework table.	

					Table 1. Secure Software Development Framework (SSDF) Version 1.2	The PDF table was not possible to copy text within one cell, making it impossible to copy-paste out single practices.	
(Independent)	Mike Sanderson	E	322				
(Independent)	Mike Sanderson	E	322	PO.1.1	PO.1.1: This is something that does not need to be done at the individual organizational or project level. It should be done at the "community" level (or the professional level).		
(Independent)	Mike Sanderson	E	322	PO 1.3	PO 1.3 should have a corresponding "solicit" for software vendors		
(Independent)	Mike Sanderson	E	322	PO 2.1	PO 2.1 This is not something that needs to be done organizational specific, but should have standard roles for standard industry titles, such as junior, senior, lead. These then can be incorporated into professional job descriptions.		
(Independent)	Mike Sanderson	E	322	PO. 2.1	PO. 2.1 "Example 7: Designate a group of individuals or a team as the code owner for each project." Teams should already have this. It would be better to declare this person is responsible for the security.		
(Independent)	Mike Sanderson	E	322	PO 2.3	PO 2.3 - No upper management won't say they want their software to be secure. The issue is in the breach. Even in highly secure settings, upper management doesn't want to pull a release because it crashes because some mid-level developer coded it to crash rather than enter an undefined state, if that state is in fact extremely low risk.		
(Independent)	Mike Sanderson	E	322	PO 3.1	PO 3.1 - Like PO 1.1, this is something that can be done at the level of industry, sector, or platform.		
(Independent)	Mike Sanderson	E	322	PO. 3.3	PO. 3.3 Notional Implementation Example "Use existing tooling" is excellent. This notes that the requirements should be included on the tickets. "Ticket writing" is a distinct step as part of the agile planning process, in which the project management and the technical team (ideally the entire technical team, but always the technical leads) break down tasks into individuals tickets that can be tracked, assigned, completed and evaluated.		
(Independent)	Mike Sanderson	E	322	PO. 4.1	P.O. 4.1 A single organization has differing requirements for each project. This phrasing seems to imply that every organization should do this for the entire SDLC. Instead, this is a risk-assessment		
(Independent)	Mike Sanderson	E	322	PO 5.1	PO 5.1 - Example 4 is the best. Example 1 is so boilerplate — use multi-factor authentication. Examples 5 and 10 are much stronger.		
(Independent)	Mike Sanderson	E	322	PO. 5.2	PO. 5.2 This is extremely unrealistic. The entire point of Development endpoints is that they do not have the hardening that production endpoints have. While it is not impossible that an unsecured development endpoint could leak information about how they system operates, information that could be useful to an adversary,		
(Independent)	Mike Sanderson	E	322	PO. 6.1	PO. 6.1 This is entirely a function of the knowledge, and so is a function of staying up-to-date and of continual training and education.		
(Independent)	Mike Sanderson	E	322	PO.6.2:	PO.6.2: "Identify new processes, tools, and techniques that can help avoid software errors (see PW.7, RV.3.3)." This is hopelessly vague.	It would be better to simply suggest that developers should be aware of new processes, tools, and techniques, and understand how they can improve security.	
(Independent)	Mike Sanderson	E	322	PS. 1.1	PS. 1.1 Example 1: Store all source code and configuration as code in a code repository..." Not All! Sensitive secretes should definitely not be stored in the repository.	Cut all, and specify NOT to store secrets in a code repository.	
(Independent)	Mike Sanderson	E	322	PW. 9.1 and 9.2	PW. 9.1 and 9.2 These are a function of training.		
(Independent)	Mike Sanderson	T	395	References	This link is to the "releases" page, not to the document being cited, and the releases page has other documents. It should link directly to the version cited.		