The Software Assurance Forum for Excellence in Code[1] (SAFECode) is pleased to have this opportunity to comment in response to the NIST Request for Information in support of the development of a cybersecurity framework. Our comments are submitted in response to the questions in Group B of the NIST RFI, and in particular to the questions about existing approaches that apply across sectors. Our members have all made major investments in their development processes which are aimed at improving the security of the software they produce. Security is important to all organizations who use software to process critical information and to manage critical business processes. Real-world experience has shown that having a secure development process is the most effective way to improve software security.

Improving the security of the software, hardware, and services that underpin critical infrastructure (CI) and critical infrastructure at greatest risk (CIGR) can reduce cybersecurity risks to vital national services. Creating a cybersecurity framework that identifies relevant international standards and best practices and still maintains flexibility for security innovation should be a central focus for NIST's efforts. The owners and operators of CI, CIGR, and the technology vendors that support them need a stable framework that is extensible, practicable, and transparent.

Government has a particular interest in improving the security of software, hardware, and services acquired for government and CI and CIGR applications. Numerous proposals intended to support this goal have been floated recently. Because SAFEcode members are committed to advancing effective software assurance methods, we thought it might be helpful to share a technical perspective on a few of the approaches contemplated in these proposals and suggest how such proposals, if crafted appropriately, can help advance cybersecurity.

Secure software development is a process and its implementation varies from one organization to another depending on the organization's security maturity, technical skills and culture. Therefore, when we see potential requirements that are highly prescriptive and/or technical, we carefully review them for unintended consequences. Mandating specific practices or tooling could inadvertently increase costs for government and industry without actually reducing risk; furthermore, such mandates could stifle the innovation necessary to counter existing and emerging threats. Rather than mandating specific practices, we believe that governments seeking additional assurance about the software they acquire should focus on vendors' secure development processes, leveraging global standards.

In our review of recent proposals, SAFECode members have seen a variety of prescriptive requirements, including:

- Requirements for the use of specific secure coding practices and creation of unique country-specific secure coding standards as well as the use of independent code assessment organizations to evaluate compliance with standards.

[1] SAFECode is a non-profit organization exclusively dedicated to increasing trust in information and communications technology products and services through the advancement of effective software assurance methods. SAFECode is a global, industry-led effort to identify and promote best practices for developing and delivering more secure and reliable software, hardware and services. Its members include Adobe, EMC Corporation, Intel Corporation, Microsoft Corp., SAP AG, Siemens AG, and Symantec Corp.

- Requirements for the use of automated static or dynamic code analysis, including specific "qualified" static analysis tools and authorized third party testing organizations to complete the analysis.
- Requirements for vendors to provide the government with source code to assess functionality, quality and security.
- Requirement for vendors to declare the absence of security vulnerabilities in their products.

Some of these proposed requirements appear to be the same as or similar to practices that SAFECode members apply and have documented in our Fundamental Practices for Secure Software Development (2nd Edition).[1] But there are key differences between what SAFECode members – and other organizations that are committed to software security – actually do and how they do it and the proposed requirements. In order to help to explain SAFECode members' concerns, we have summarized these differences below.

- Studies have repeatedly shown that complex software has errors (bugs) and a small number of these errors are security vulnerabilities. A strong software engineering process can be directly linked to the reduction of these errors, but even the most mature software engineering organizations do not completely eliminate software errors. For this reason, any vendor claiming the absence of vulnerabilities in the code it ships would be only demonstrating its lack of understanding of the field of software assurance.
- Coding standards are an important element of a secure development process because they help our developers use programming languages in safe ways, and we can automatically detect deviations from those standards and tell the developers to correct them. But coding standards vary because our members use different compilers, different operating system platforms and versions, and build software that's used for different purposes. If our members had to comply with a "one size fits all" government secure coding standard, the members would spend time recoding to the standard – instead of focusing on the security needs of their code bases and improving the real security of the software.

As part of our organizations' software assurance processes, SAFECode members also have used outside organizations (contractors and consultants) to help with testing, evaluation and certification. They are able to effectively guide contractors' efforts because they understand their code bases. But if the U.S. government takes national approaches to standards and testing, other countries will likely follow. The result could have negative consequences on the global technology marketplace and slow innovation, including advances in software security. For example, IT vendors would have to train personnel at multiple different outside code assessment centers, diverting internal security experts from making software more secure. Effort by developers on making their software more secure will have much greater payoffs than accommodating individual assurance schemes each of which takes its own national perspective. Wherever possible, the cybersecurity framework that NIST develops should support international standards that have been demonstrated to be effective in reducing or managing risk.

SAFEcode white papers, such as the Fundamental Practices paper discussed above, acknowledge that static analysis is an important practice as part of a secure development process and SAFECode members all use static analysis tools. But we use a variety of different tools, often more than one, and some companies use "home grown" tools or tool extensions. Regardless of the tool or tools used, every company has to "tune" or tailor them to find software vulnerabilities while minimizing the number of "false positives" that the tools emit. These practices – tuning the tools, extending the tools – are required because each of us has a different code base with different technical attributes. If policy

proposals or procurement requirements mandate the use of government “qualified” tools, some of our tools surely won’t be on the “qualified” list. As a result, we’ll still have to use our own tools, tuning and extensions to improve security, and also run the “qualified” tools and then manage the increased number of false positives they produce – which will incur more cost and extend product cycles.

A final point concerns the lifecycle of security requirements. While many of the proposals we are seeing refer to static standards and tooling, the reality for all of us is that security needs evolve and how we implement the various software assurance practices must do so as well. SAFECode members assess their development processes, including vulnerabilities identified after release, to determine when and how additional practices or variations in implementation may be needed to improve security. For example, it’s not atypical for one of our members to update its secure development process – static analysis tuning and extensions, coding standards, and other requirements – once or twice a year. Government-mandated tools and standards simply could not keep up with that pace. Better software assurance is delivered through a comprehensive development process that can evolve and adapt.

Each of the SAFECode members has its own secure development process. Our processes have common elements, and those elements were the basis for the Fundamental Practices document. We find it interesting to see some of the 17 practices and principles outlined in our Fundamental Practices paper included in various policy proposals. But no one practice is a silver bullet to provide better software assurance. Rather, the efficacy and efficiency of each of these practices varies based on how they are applied as part of a holistic process within each unique organization developing software.

If the US government seeks additional assurance about acquired software it should work with vendors to understand the processes they use to develop software. Emerging global standards, such as ISO 27034-1, may also offer a way to verify that an organization has a software development process and is following it. Prescriptive mandates for specific practices, coding standards, tools or consultants could make vendors’ processes impractical or, worse, have effects counter to their original intent. Such mandates will not provide the flexibility necessary to address today’s complex threats nor to innovate in the face of a rapidly changing threat landscape. On the other hand, asking vendors to document their security development processes will encourage the right behavior on the part of vendors and their developers and enable them to innovate over time. Knowing, and, as appropriate, verifying that vendors have a process for addressing security in the development process will give acquirers additional confidence in the software on which they depend.

In addition to secure development of software, SAFECode has done substantial work on the issues of supply chain security, process metrics, and security engineering training. We believe that these efforts represent best practices that can also inform aspects of the cybersecurity framework. We have attached a summary of these efforts and links to the materials in Appendix A.

Thank you for giving us the opportunity to comment on the direction of NIST's efforts to develop a cybersecurity framework. If you have any questions, or would like to discuss these comments, please contact us through stacy@safecode.org.

Sincerely,

Steven B. Lipner
Chairman
www.safecode.org

Appendix A

Guidance for Agile Practitioners

SAFECode Releases Software Security Guidance for Agile Practitioners
This paper provides practical software security guidance to Agile practitioners in the form of security-focused stories and security tasks they can easily integrate into their Agile-based development environments. SAFECode has also made available quick reference guides from the paper for download.

http://www.safecode.org/publications/SAFECode_Agile_Dev_Security0712.pdf 1.5M
http://www.safecode.org/publications/SAFECode_Agile_Section2b-tables.pdf 735K
http://www.safecode.org/publications/SAFECode_Agile_Section3-tables.pdf 730K
http://www.safecode.org/publications/SAFECode_Agile_Section2a-tables.pdf 1.4M

Interpreting the BSIMM

A SAFECode Perspective on Leveraging Descriptive Software Security Initiatives
This brief paper provides SAFECode's perspectives on the BSIMM and addresses the questions that we often get about how our guidance relates to the data released through the BSIMM effort.

http://www.safecode.org/publications/SAFECode_Interpret_BSIMM1111.pdf 788K

Fundamental Practices for Secure Software Development 2nd Edition

Report Provides Foundational Set of Secure Development Practices Based on an Analysis of the Real-World Actions of SAFECode Members. The report is intended to help others in the industry initiate or improve their own software security programs and encourage the industry-wide adoption of fundamental secure development methods.

http://www.safecode.org/publications/SAFECode_Dev_Practices0211.pdf 1.9M

Overview of Software Integrity Controls

An Assurance-Based Approach to Minimizing Risks in the Software Supply Chain. The new report provides actionable recommendations for minimizing the risk of vulnerabilities being inserted into a software product during its sourcing, development and distribution.

http://www.safecode.org/publications/SAFECode_Software_Integrity_Controls0610.pdf 2.3M

Framework for Software Supply Chain Integrity

First industry-driven framework for analyzing and describing the efforts of software suppliers to mitigate the potential that software could be intentionally compromised during its sourcing, development or distribution.

http://www.safecode.org/publications/SAFECode_Supply_Chain0709.pdf 1.4M

Security Engineering Training

A Framework for Corporate Training Programs on the Principles of Secure Software Development

http://www.safecode.org/publications/SAFECode_Training0409.pdf 1.9M