5

Wait, What?! DoD Access to Source Code

A Defense Innovation Board (DIB) Backgrounder

Richard Murray, Trae' Stephens, Michael McQuade, Milo Medin, Gilman Louie

Version 1.2, 28 Oct. 2019

Executive Summary

In the Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) report[1] we made the recommendation that DoD "require access to source code, software frameworks, and development toolchains – with appropriate IP rights – for DoD-specific code, enabling full security testing and rebuilding of binaries from source". This recommendation (D1) has generated substantial discussion in industry and, in some cases, has been interpreted to mean that DoD should only acquire software if it comes with source code rights. We continue to support this recommendation but since the recommendations will need to be applied in different circumstances ("not all software is the same") some additional discussion is perhaps useful.

The SWAP report discusses the background and motivation behind this recommendation and provides numerous examples of its potential use. In this (short) concept paper, we try to pull together the various threads in the report related to this specific recommendation to provide more clarity on how this might be applied in the context of DoD acquisition of software and software intensive systems. Our primary points were and remain:

1. Whenever possible, DoD should use existing commercial software to solve problems that are common between DoD and commercial uses, including modifying DoD processes to match commercial processes to take advantage of the scale, maturity, and speed of commercial software. Easy examples are software for word processing, travel planning, inventory, and audit. In some cases, a combination of commercial software with custom modules may be the answer (for example, for health care records, where 90+% of the uses cases are standard but a custom module for combat/field operations may be needed).
2. Whenever feasible and useful, DoD should seek to obtain source code for non-custom software ("Type A") for the purposes of vulnerability scanning and related analyses. The utility of this will depend on the application (data analysis applications running on highly classified databases might be need evaluated for consistency with a Zero Trust Architecture by examination of source code, for example). Having said this, it is likely that commercial code that is widely deployed is going to be less vulnerable than DoD-specific code that is used only within DoD => default to option 1 (use COTS when possible, with process modifications as needed) versus contracting out to rewrite code that serves the same purpose but is highly "tuned" to DoD idiosyncrasies.
3. Every *purpose-built* DoD software system should include source code as a deliverable (commandment #6[2]).

[1] *Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage,* DIB, 3 May 2019.
[2] *Defense Innovation Board Ten Commandments of Software*, 3 May 2019.

The Defense Innovation Board (DIB) Software Acquisition and Practices (SWAP) report and supporting concept papers provide multiple comments on the importance of access to source code and the need to use commercial software when possible. We extract these comments here and add additional clarifying comments (in blue).

Chapter 1. Who Cares: Why Does Software Matter for DoD?

1.2 Weapons and Software and Systems, Oh My! A Taxonomy for DoD

We define three broad operational categories:

- *Enterprise systems*: very large-scale software systems intended to manage a large collection of users, interface with many other systems, and generally used at the DoD level or equivalent. These systems should always run in the cloud and should use architectures that allow interoperability, expandability, and reliability. In most cases the software should be commercial software purchased (or licensed) without modification to the underlying code, but with DoD-specific configuration. Examples include: e-mail systems, accounting systems, travel systems, and HR databases.

- *Business systems*: essentially the same as enterprise systems, but operating at a slightly smaller scale (e.g., for one of the Services). Like enterprise systems, they are interoperable, expandable, reliable, and probably based on commercial offerings. Similar functions may be customized differently by individual Services, though they should all interoperate with DoD-wide enterprise systems. Depending on their use, these systems may run in the cloud, in local data centers, or on desktop computers. Examples include: software development environments, Service-specific HR, financial, and logistics systems.

- *Combat systems*: [omitted]

Having defined systems that deliver effects and the kinds of computing platforms on which software is hosted, we now distinguish between four primary types of software, which we use throughout the rest of the report so that we differentiate the acquisition and deployment approaches that are needed:

- **Type A (Commercial-Off-The-Shelf [COTS] apps):** The first class of software consists of applications that are available from commercial suppliers. Business processes, financial management, human resources, software development, collaboration tools, accounting software, and other “enterprise” applications in DoD are generally not more complicated nor significantly larger in scale than those in the private sector. Unmodified commercial software should be deployed in nearly all circumstances. Where DoD processes are not amenable to this approach, those processes should be modified, not the software.

The SWAP report emphasizes the fact that not all software is the same and different types of software require different acquisition and development processes. In particular, for “commercial software” (as outlined in the examples above), it is generally preferable for DoD to use existing applications and adopt its processes to allow industry standard practices as much as possible.

This has the advantage that widely deployed commercial software is likely to be optimized for efficiency and has a higher likelihood of being secure to common cyberattacks since commercial code is under constant attack and vulnerabilities are fixed quickly.

Chapter 2. What Does It Look Like to Do Software Right?

2.1 How It Works in Industry (and Can/Should Work in DoD): DevSecOps

Software development. These are software engineering practices that include source code management, software build, code review, testing, bug tracking, release, launch and post-mortems. Some of the key best practices that are applicable to DoD software programs include:

- All source code is maintained in a single repository that is available to all software engineers. There are control mechanisms to manage additions to the repository but in some cases all engineers are culturally encouraged to fix problems, independent of program boundaries.
- Developers are strongly encouraged to avoid “forking” source code (creating independent development branches) and focus work on the main branch of the software development.
- Code review tools are reliable and easy to use. Changes to the main source code typically require review by at least one other engineer, and code review discussions are open and collaborative.
- Unit test is ubiquitous, fully automated, and integrated into the software review process. Integration, regression, and load testing are also widely used and these activities should be an integrated automated part of daily workflow.
- Releases are frequent—often weekly. There is an incremental staging process over several days, particularly for high-traffic, high-reliability services.
- Post-mortems are conducted after system outages. The focus of the post-mortem is on how to avoid problems in the future and not about affixing blame.

For those instances where DoD is developing code (either organically, with contractors, or via contracts), source code should be part of the deliverable for the project and should be managed according to commercial practices, as outlined above. These activities make clear that source code access is required.

Chapter 4. How Do We Get There from Here: Three Paths for Moving Forward

4.1 Path 1: Make the Best Out of What We’ve Got

The following list provides a summary of high-level steps that require changes to DoD culture and process, but could be taken with no change in current law and relatively minor changes to existing regulations:

- Require access to source code, software frameworks, and development toolchains, with appropriate intellectual property (IP) rights, for all DoD-specific code, enabling full security testing and rebuilding of binaries from source.

4.2 Path 2: Tune the Defense Acquisition System to Optimize for Software

The following list provides a set of high-level steps that require some additional changes to DoD culture and process, but also modest changes in current law and existing regulations. These steps build on the steps listed in path 1 above, although in some cases they can solve the problems that the previous actions were trying to work around.

- Create streamlined authorization and appropriation processes for defense business systems (DBS) that use commercially-available products with minimal (source code) modification.
- For any software developed for DoD, require that software development be separated from hardware in a manner that allows non-prime vendors to bid for software elements of the program on a performance-based basis.

Chapter 4 reviews some of the options for moving forward, with additional details on those actions we recommend described in more detail in Chapter 5.

Chapter 5. What Would the DIB Do: Recommendations for Congress and DoD

Line of Effort D. DoD and industry must change the practice of how software is procured and developed by adopting modern software development approaches, prioritizing speed as the critical metric, ensuring cybersecurity is an integrated element of the entire software lifecycle, and purchasing existing commercial software whenever possible.

Recommendation D1. Require access to source code, software frameworks, and development toolchains, with appropriate IP rights, for all DoD-specific code, enabling full security testing and rebuilding of binaries from source

For many DoD systems, source code is not available to DoD for inspection and testing, and DoD relies on suppliers to write code for new compute environments. As code ages, suppliers are not required to maintain codebases without an active development contract and “legacy” code is not continuously migrated to the latest hardware and operating systems. The desired state is that DoD has access to source code for DoD-specific software systems that it operates and uses to perform detailed (and automated) evaluation of software correctness, security, and performance, enabling more rapid deployment of both initial software releases and (most importantly) upgrades (patches and enhancements). DoD is able to rebuild executables from scratch for all of its systems, and has the rights and ability to modify (DoD-specific) code when new conditions and features arise. Code is routinely migrated to the latest computing hardware and operating systems, and routinely scanned against currently-known vulnerabilities. Modern IP language is used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors are able to use licensing agreements that protect any IP that they have developed with their own resources. Industry trusts DoD with its code and has appropriate IP rights for internally developed code.

Recommendation D1 is the primary recommendation regarding source code. As we make clear, the intent is that DoD-specific code be available for security testing and rebuilding of binaries from

source. This access is usually not needed for commercial source code and our recommendation does not address access to source code for commercial software.

In the draft implementation plan (Appendix A of the report), we provide some possible actions to implement this recommendation:

	Draft Implementation Plan	Lead Stakeholders	Target Date
D1.1	Work with industry to modernize policies for software code ownership, licensing, and purchase. See 2018 Army IP directive as an example.	USD(A&S)	Q3 FY19
D1.2	Modify FAR/DFARS guidance to require software source code deliverables for GOTS and for government-funded software development. Obtain rights for access to source code for COTS wherever possible (and useful)..	USD(A&S)	Q3 FY20
D1.3	Modify DoDI 5000.02 and DoDI 5000.75 to make access to code and development environments the default.	USD(A&S)	Q3 FY20
D1.4	Develop a comprehensive source code management plan for DoD including the safe and secure storage, access control, testing and field of use rights.	USD(A&S), with CIO	Q4 FY20

These actions all point to the need for the government to work with industry to provide a workable approach to providing access to source code. The 2018 Army IP directive provides a good starting point for this effort. We note in particular the following language from that report regarding different types of software and how it should be treated:

> *"[Software] developed by a contractor exclusively at private expense: The contractor may restrict the right of the Government to release or disclose technical data to persons outside the Government or permit such persons to use the technical data." (c1(b))*

We do believe that there are instances when commercially available code may be used in a setting in which the threat model or security requirements are sufficiently different from commercial usage that DoD may want to perform additional testing of the code. This could be done by licensing access to source code or by making use of a trusted third party to carry out vulnerability testing.

Thus, whenever feasible and *useful*, we believe that DoD should seek to obtain source code for non-custom software ("Type A") for the purposes of vulnerability scanning and related analyses. The utility of this will depend on the application (data analysis applications running on highly classified databases might be need evaluated for consistency with a Zero Trust Architecture by examination of source code, for example). Having said this, it is likely that commercial code that is widely deployed is going to be less vulnerable than DoD-specific code that is used only within DoD. Hence the default should be to use COTS when possible (with process modifications as needed) versus contracting out to rewrite code that serves the same purpose but is highly "tuned" to DoD idiosyncrasies.

5.4 Kicking the Can Down The Road: Things That We Could Not Figure Out How to Fix

Using commercial software whenever possible. DoD should not build something that it can buy. If there is an 80 percent commercial solution, it is better to buy it and adjust—either the

requirements or the product—rather than build it from scratch. It is generally not a good idea to over-optimize for what we view as "exceptional performance," because counter-intuitively this may be the wrong thing to optimize for as the threat environment evolves over time. Similarly, actions should be taken to ensure that the letter and spirit of commercial preference laws (e.g., 10 USC 2377, which requires defense agencies to give strong preference to commercial and non-developmental products) are being followed.

This section of the report speaks directly to the preference for using commercial software solutions when available. There is no intent in the report's recommendations to make access to source code a determining factor in the acquisition of a COTS software solution. If a COTS software solution is available and most suited for the needed capability but using it may come without source code access, the COTS solution will likely still be preferred. When source code access is available and/or needed, the proper processes for storage, handling, and scanning of source code for vulnerability assessment must be established. It may make sense to manage this via a third party rather than by each individual program office.

Appendix D. Frequently Asked Questions (FAQs)

6. Providing source code to the government is a non-starter for industry. How will they make money if they have to give the government their code?

It is critical that DoD have access to source code for purpose-build software: it is required in order to do security scans to identify and fix vulnerabilities, and only with access to the source code and build environment can the government maintain code over time. However, providing source code is different than handing over the rights to do anything they want with that code. Modern intellectual property (IP) language should be used to ensure that the government can use, scan, rebuild, and extend purpose-built code, but contractors should be able to use licensing agreements that protect any IP that they have developed with their own resources.

Appendix E. DIB Guides for Software

Ten Commandments of Software

Commandment #6. Every purpose-built DoD software system should include source code as a deliverable. DoD should have the rights to and be able to modify (DoD-specific) code when new conditions and features arise. Providing source code will also allow the DoD to perform detailed (and automated) evaluation of software correctness, security, and performance, enabling more rapid deployment of both initial software releases and (most importantly) upgrades (patches and enhancements). [Types C, D]

Supporting recommendation: Use commercial process and software to adopt and implement standard business practices within the services. Modern enterprise-scale software has been optimized to allow business to operate efficiently. The DoD should take advantage of these systems by adopting its internal (non-warfighter specific) business processes to match industry standards, which are implemented in cost-efficient, user-friendly software and

software as a service [SaaS] tools. Rather than adopt a single approach across the entire DoD, the individual services should be allowed to implement complementary approaches (with appropriate interoperability).

The "Ten Commandments" emphasized the fact that not all software is the same, and categorized software into four main types:

- A: commercial ("off-the-shelf") software with no DoD-specific customization required;
- B: commercial software with DoD-specific customization needed;
- C: custom software running on commodity hardware (in data centers or in the field);
- D: custom software running on custom hardware (e.g., embedded software).

Commandment #6 focused on Types C & D (*custom* software).

DIB Metrics for Software

#	Metric	Target value (by software type)				Typical DoD values for SW
		COTS apps	Custom -ized SW	COTS HW/OS	Real-time HW/SW	
9	% code avail to DoD for inspection/rebuild	N/A	100%	100%	100%	?

As in the 10 commandments, this table called out the need to provide source code access for *custom* software. We note that there is a slight discrepancy here from the "Ten Commandments" document, in that this table identifies software of Type B ("customized") software as requiring source code access. This would likely be restricted to the customized portions of an application (which are again DoD specific).

Do's and Don'ts for Software

Observed practice (Don'ts)	Desired state (Do's)	Potential Barriers
Require customized software solutions to match DoD practices	For common functions, purchase existing software and change DoD processes to use existing apps	Culture
Depend almost entirely on outside vendors for all product development and sustainment	Require source code as a deliverable on all purpose-built DoD software contracts. Continuous development and integration, rather than sustainment, should be a part of all contracts. DoD personnel should be trained to extend the software through source code or API access	Culture (no apparent statutory obstacle) FAR/DFARS technical data rights

Business processes, financial, human resources, accounting and other “enterprise” applications in the DoD are generally not more complicated nor significantly larger in scale than those in the private sector. Commercial software, unmodified, should be deployed in nearly all circumstances. Where DoD processes are not amenable to this approach, those processes should be modified, not the software. Doing so allows the DoD to take advantage of the much larger commercial base for common functions (e.g., Concur has 25M active users for its travel software).