NOT FOR PUBLICATION UNTIL RELEASED BY
HOUSE SCIENCE, SPACE, AND TECHNOLOGY COMMITTEE
SUBCOMMITTEE ON INVESTIGATIONS AND OVERSIGHT
SUBCOMMITTEE ON RESEARCH AND TECHNOLOGY
UNITED STATES HOUSE OF REPRESENTATIVES

DEPARTMENT OF THE AIR FORCE PRESENTATION TO THE HOUSE COMMITTEE ON SCIENCE, SPACE, AND TECHNOLOGY SUBCOMMITTEE ON INVESTIGATIONS AND OVERSIGHT, AND SUBCOMMITTEE ON RESEARCH AND TECHNOLOGY UNITED STATES HOUSE OF REPRESENTATIVES

HEARING DATE/TIME: May 11, 2022 10:00 A.M.

SUBJECT: Securing the Digital Commons: Open-Source Software Cybersecurity

STATEMENT OF:

Ms. Lauren Knausenberger
Chief Information Officer
Office of the Secretary of the Air Force

NOT FOR PUBLICATION UNTIL RELEASED BY
HOUSE SCIENCE, SPACE, AND TECHNOLOGY COMMITTEE
SUBCOMMITTEE ON INVESTIGATIONS AND OVERSIGHT
SUBCOMMITTEE ON RESEARCH AND TECHNOLOGY
UNITED STATES HOUSE OF REPRESENTATIVES

Introduction & Strategic Implications

Less than five years passed from the formal establishment of the Manhattan Project to the operational use of a nuclear weapon, at the time an unprecedented acceleration of research, development, and operationalization of a new capability. A future conflict may be decided by features, updates, or fixes to software-intensive systems that take place in hours or even minutes, not days, much less years. In order to meet the strategic imperative this moment requires, the Air Force must lower the barrier to fielding new algorithms and systems while increasing the utilization of commercial software, including open-source investments, to diversify the innovation space for the Department.

Open-source software (OSS) is software that has the source code made publicly available, is licensed for broad public reuse, and is typically distributed free of charge. By leveraging open-source software, the development of large-scale software projects can be drastically accelerated due to the re-use of previously developed code and the open systems architectures that open-source software tends to drive. In order to take maximum advantage of the opportunities provided by open-source software, entities in charge of software development efforts should take steps to mitigate some of the risks associated with open-source software.

OSS should not be confused with so-called “shareware,” “freeware,” or “freemium” software products that are distributed free of charge but require the user to pay to unlock some additional functionality or to continue using the software after a trial period. That’s not to say that there are not successful business models built around open-source software: any software requires support and maintenance, and some companies have made a robust business in providing support services for free and open-source software. As an example, Red Hat Enterprise Linux (RHEL) is a free and open-source product for which Red Hat provides paid, enterprise-class support.

Background

Software Life Cycle – Development, Distribution, and Maintenance

The life cycle of an open-source software product can broadly be divided into development, distribution, and maintenance. Development refers to the specification of what the software should do and writing the code to implement the desired functionality; distribution refers to the process of packaging the software appropriately and providing it to the end users (which may be software developers in the case of software dependencies); and maintenance refers to the process of fixing errors (frequently referred to as “bugs”), updating software to maintain compatibility with other software and hardware systems, and incorporating any updates to the dependencies for that software. Due to the constantly evolving nature of modern agile software these life-cycle phases frequently overlap significantly, with features regularly being added to software that is continuously distributed and maintained.

Software Dependencies

A piece of software that is incorporated into a larger piece of software is referred to as a *dependency* of the larger piece of software. Each dependency may in turn have other dependencies (which may in turn have dependencies...) The total collection of dependencies and dependencies-of-dependencies is referred to as the set of *transitive dependencies*, and large software projects may have thousands of transitive dependencies.

Open-source software may be distributed on its own (as is the case with the Android Open Source Project (AOSP, commonly just “Android”) mobile operating system that powers Android smartphones) or as part of a proprietary product (Apple’s proprietary iOS mobile operating system that powers the iPhone

platform is based on the open-source BSD operating system). In that case, the open-source software is a dependency of the commercial product.

Because of the way that software is constructed out of many layers of dependencies, even if only proprietary software is purchased there are likely still open-source software dependencies and therefore open-source software supply chains that must be considered.

Software Risks

Incorporating open-source software into a software project can significantly speed up the delivery and decrease the cost of developing that software due to re-use of previously developed software. It would be difficult to name a successful software project today that didn't rely on OSS in some form.

Successfully using open-source software requires being cognizant of certain risks:

- The incorporation of any software dependencies (whether open- or closed-source) into your project can expose you to responsibility for maintaining that dependency during the lifetime of your project. If the company or team that developed the software decides to end support for the product, then users may be forced to find a replacement product or pay for an expensive one-off support contract for maintenance. This maintenance also includes the correction of security deficiencies as they are discovered; for this reason, the use of unmaintained software poses both a business and cyber security risk.
    - Example: For years after Windows XP was formally discontinued, Microsoft continued to support existing Windows XP customers via support contracts that increased in price over time (due to a smaller and smaller customer base to distribute the cost across).
- Any software dependency (open-source or proprietary) may inadvertently introduce security flaws into your software product. These broadly fall into the categories of known flaws (these are typically published as Common Vulnerabilities and Exposures, or CVEs) and as-yet-undiscovered flaws (known as "zero-days" in the cybersecurity community). Security-conscious organizations such as the DoD may have more stringent requirements for mitigating these flaws than other organizations, making it difficult for them to utilize the dependency because of a lack of evidence supporting the security of the dependency, disagreement over the severity of the flaws in the software, or timelines for mitigations of flaws that don't satisfy policy or operational requirements.
    - Example: Several months ago, several severe zero-day vulnerabilities in an open-source logging framework named "log4j" were discovered and eventually disclosed as CVEs; thousands or millions of software projects needed to incorporate newer versions of log4j into their projects quite rapidly in order to avoid falling victim to cyber attacks.
- The distribution channels for software may themselves come under attack – this is what is commonly referred to as a "supply chain attack." These sorts of attacks come in several forms. A malicious developer may decide to publish a version of the software that is intentionally flawed, and users that automatically update to the latest version may suffer a degradation in the capability of their system as a result. Malicious developers may publish copy-cat software using a similar name to a common piece of software that is malicious software, hoping that a user will accidentally download and use their malicious software. Finally, if a distribution channel is insufficiently hardened against attack, an adversary may be able to make changes to software in the distribution channel without the developer or using realizing it.
    - Example: The author of a popular open-source package named "left-pad" grew disgruntled with the open-source community and intentionally deleted the software from

the distribution service, briefly causing outages of several prominent websites and probably thousands of less prominent ones.
- Example: The internal development environment at SolarWinds was compromised in a sophisticated supply chain attack that introduced malicious code into their proprietary (i.e. not OSS) software product, which in turn exposed the networks of dozens of governmental and commercial entities to attack.

Software Policy

The documents listed below are a few of the governing policies that relate to the secure use of open-source software by the Air Force. They are not intended to be comprehensive.

Executive Order 14028, Improving the Nation's Cybersecurity (May 12, 2021)

This executive order mandates several efforts to improve the cybersecurity posture of the country. Of particular note to the topic of software dependencies is section 4, "Enhancing Software Supply Chain Security." This section mandates, among other things, the use of secure build environments for compiling software source code into usable programs and the creation of a Software Bill of Materials (SBOM) that is distributed with software that is utilized by the Federal government. This SBOM will assist in knowing what software dependencies exist within a project in order to accurately assess the overall risk of using that project, and to aid in the long-term maintenance and operations of the software. In the log4j example above, the Air Force's use of SBOMs, described below, allowed for the rapid identification of systems that required an updated version of log4j.

DoD CIO Memo on Software Development and Open-Source Software (Jan 24, 2022)

This memorandum expands upon and clarifies a 2009 memo on open-source software use in the DoD. Key points included are that the use of open-source software is explicitly allowed in the DoD, that DoD personnel may contribute code to open-source software as part of their regular duties, and that the DoD must actively manage software supply-chain risk in accordance with the 2018 DoD Cyber Strategy.

DoD Development, Security, and Operations (DevSecOps) Reference Design

The DevSecOps reference design outlines the technology, people, and procedures necessary to create a secure environment for developing and operating secure software. Software development organizations within the Air Force look to the reference design as a starting point for standing up a software development center.

Platform One Efforts to Enhance Software Supply Chain Security

Platform One is an Air Force organization that serves as enterprise provider of development, security, and operations tools and services for the DoD. These services include a secure gateway between secure government cloud environments and the commercial internet; a managed platform-as-a-service for developing and deploying cloud-based applications; the Iron Bank hardened container repository, and the Big Bang implementation of the DoD DevSecOps reference design.

Iron Bank (DoD Hardened Container Repository)

Platform One's Iron Bank is a repository for container images, which is one of the most popular software packaging formats for both OSS and proprietary software that is designed to run in the cloud. Containers are a "cloud native" method of packaging containers and are widely used at Google, Amazon, Microsoft,

Oracle, and virtually every organization that provides or hosts cloud-based software, including Platform One. Iron Bank adds several levels of security that today mitigate some of the risks mentioned above:

- When projects are onboarded, developer identities are verified, reducing the risk of a malicious developer intentionally inserting malicious code into a codebase.
- Copy-cat project names that might confuse users or developers into using the wrong project are not allowed.
- Automated security checks and validation by cybersecurity experts enforce best practices in the development of software containers, potentially reducing the severity of any "zero-day" vulnerabilities that may exist inside of that container.
- Software is compiled and packaged using secure "pipelines" that run inside of the government-controlled software environment, protecting against the introduction of malicious code during the packaging process.
- Software Bills of Materials (SBOMs) are generated for every container in the repository, allowing organizations using those containers to build an accurate, up-to-date inventory of software (including all the dependencies) running in their environment.
- Software in the repository is continuously scanned and compared to the most up-to-date list of Common Vulnerabilities and Exposures (CVEs), so that when a vulnerability is discovered it can be remediated in a timely fashion.
- The Iron Bank repository is hosted in a secure cloud environment and is continuously monitored and assessed according to DoD cyber security standards, ensuring the integrity and availability of the repository and securing it from attack by our adversaries.

Iron Bank has been a successful venture to date: there are approximately 1,000 containers available on Iron Bank, each of which has been assessed against DoD criteria for hardened containers. In the near term, Platform One is focused on improvements to allow Iron Bank to scale in an agile fashion. These include publishing fully automated assessments of a container image to provide fast, objective feedback to DoD organizations that are considering using a piece of software from Iron Bank (a "nutrition label" for a software container), increased use of digital signatures to validate the entire software supply chain, and automated notifications to users (on an opt-in basis) to notify them if they may be using a piece of software that has had a vulnerability disclosed. Because it is a centralized repository, Platform One can search across the repository for critical vulnerabilities (such as the log4j vulnerabilities) and notify anyone that is impacted by the vulnerability (this search is enabled by SBOMs, which provide an inventory of the software in a system). On occasion, the Iron Bank team has provided the first notification to OSS developers that one of their dependencies is vulnerable and needs to be updated.

Big Bang (DevSecOps Reference Design Implementation)

Secure software operations depend on both secure software and correct configuration of that software. Platform One provides a secure baseline for a Development, Security, and Operations (DevSecOps) under the name Big Bang. By adhering to the reference design, Big Bang provides compartmentalization of running applications to minimize the impact of any security incident, implements the services and connections to conduct active cyber defense, and runtime security to detect and neutralize threats to the applications. Big Bang also increases developer productivity and return-on-investment by providing a common baseline for developing and deploying applications. Big Bang is itself an open-source piece of software. Iron Bank (and other Platform One services) utilize Big Bang to provide a secure environment for hosting their applications.

Conclusion

Iron Bank is a successful effort to reduce some aspects of supply chain risk for both open-source and proprietary/commercial software for DoD use. Software security (like cyber security in general) is a constantly evolving and constantly improving effort, not a line of effort that can be "completed" and then focus turned elsewhere. Iron Bank will continue to evolve as better techniques for avoiding software supply-chain risk are developed and implemented. By presenting an enterprise capability for consuming containerized software, Iron Bank users can avoid "re-inventing the wheel" and unnecessarily duplicating the work that has already been done. Finally, Iron Bank and Big Bang serve as foundational capabilities for building enterprise development, security, and operations solutions.