



To: The Office of the National Cyber Director (ONCD)

8 November 2023

Re: Proposal Submission for Request for Information (RFI)


We are pleased to submit our proposal on behalf of the Open Source Security Foundation. We have thoroughly reviewed the requirements outlined in the RFI and have crafted our proposal to effectively address the needs and challenges specified. We believe that our unique skill set and deep understanding of the domain makes us an ideal partner for this endeavor.


Point of Contact (Primary): Omkhar Arasaratnam, OpenSSF General Manager

Website: <https://openssf.org/>

Email Address: omkhar@linuxfoundation.org


Phone Number: +1 646 321 6202

DocuSigned by:

8AAAEFF7FCCA496...
Arun Gupta
Vice President and General Manager for Open
Ecosystem, Intel
OpenSSF Governing Board Chair

DocuSigned by:

39419DF60AD6478...
Omkhar Arasaratnam
OpenSSF General Manager

DocuSigned by:

7059F609FFDE46B...
Brian Fox
Chief Technology Officer, Sonatype
OpenSSF Governing Board Member
Chair, OpenSSF ONCD-RFI Committee

DocuSigned by:

027DB3E0B9CC41E...
David A. Wheeler, PhD
Director of Open Source Supply Chain Security,
Linux Foundation
Facilitator, OpenSSF ONCD-RFI Committee

OpenSSF Response to the Office of the National Cyber Director's Request for Information on Open Source Software Security

The Open Source Security Foundation (OpenSSF) appreciates the opportunity to provide comments in response to the Office of the National Cyber Director's (ONCD) [Request For Information \(RFI\)](#) on open source software (OSS) security.

[OpenSSF](#) is a cross-industry organization that brings together the industry's most important open source security initiatives and the individuals and companies that support them. The OpenSSF is committed to collaboration and working both upstream and with existing communities to advance open source security for all. We have over 100 [members](#), including some of the largest tech companies, financial services, and other organizations involved in the development of OSS. Below are our responses to select topics raised in the RFI. We welcome the opportunity to continue the conversation.

Areas of Focus

OpenSSF appreciates the efforts of the federal government relating to OSS security to date, particularly through outreach such as this RFI. We recommend that the federal government, like all organizations, continue partnering with existing foundations and projects (where practical) so everyone's knowledge and resources can be pooled to maximize results as these efforts move forward. We identified these areas of priority, as described below:

1. Accelerate update of fixed components to replace components with known vulnerabilities (vulnerable log4j versions are still in wide use!)
2. Support international collaboration to harmonize worldwide regulations and policies to avoid fragmenting the OSS ecosystem
3. Align economic incentives to ensure the sustainability of secure open source software
4. Harden repositories and package managers to counter common supply chain attacks (e.g., typosquatting and dependency confusion)
5. Educate software developers to know how to develop secure software
6. Improve Incident Response
7. Invest in Research & Development (R&D) for innovation
8. Foster the adoption of memory safe programming languages, techniques, and tools

Accelerate uptake of fixed components

In the wake of high visibility events such as Log4shell, most of the focus has been on improving OSS through education, tooling, etc. This ignores the current reality that many software vendors reuse OSS without updating known-vulnerable components in a timely manner. This results in statistics such as "18 months post Log4shell, 30% of the versions consumed from Maven Central are of the known vulnerable versions". For this reason, the OpenSSF End Users Working Group created the "[Open Source Consumption Manifesto](#)" to shine a light on the fact

that companies integrating open source software into their products need to implement, at a minimum, basic hygiene measures.

We believe that the National Cybersecurity Strategy is on the right track when it talks about changing economic incentives for software publishers to focus on basic security. The strategy rightfully recognizes that software perfection is currently impossible. The strategy further calls for the definition of Safe Harbor practices that when followed would allow organizations exemption from liability. We would like to see the government work with industry and ecosystem stakeholders to define these initial standards.

Recommended actions:

- **Convene multi-stakeholder sessions to begin defining the mechanisms for Safe Harbors** as described in the National Cyber Security Strategy.

International Collaboration

The software market is global. Software is often developed by people living around the world, and software components developed in one area are reused in others. The collaboration between international partnerships will help develop a secure and harmonious global software development fabric.

Recommended action:

- **Work with others to harmonize regulations and ensure they are both effective and practical.** We ask that ONCD work with other US government partners who have remit to work with international regulators to create a regular forum where we can collaborate globally to harmonize standards and regulations to the benefit of everyone. Examples include [DHS Office of Policy \(PLCY\)](#) , [Department of State's Bureau of Cyberspace and Digital Policy \(CDP\)](#), [NIST's Office of International Affairs](#), and the [US Department of State Special Envoy for Critical and Emerging Technology \(S/Tech\)](#).

Behavioral and Economic Incentives to Secure the Open-Source Software Ecosystem

As a public good, there is a market failure when it comes to dedicating resources to open-source communities. There are few incentives for many organizations to participate, and yet those organizations all benefit when *another* organization does commit resources and personnel to the cause – a variation of the "tragedy of the commons." The government is uniquely situated to intervene in such situations to provide incentives that will encourage organizations to participate actively in open-source communities and, in particular, to remediate vulnerabilities in OSS components in a timely manner consistent with severity.

Recommended action:

- **Update VEP Process:** Eliminate or at least modernize the US government's Vulnerabilities Equities Process (VEP). Our understanding is that under the VEP, vulnerabilities found by the US government and its contractors are often deliberately withheld from software developers. This prevents fixing those vulnerabilities and leaves users open to attack. We would encourage the US to eliminate this process and share those vulnerabilities with developers so vulnerabilities can be repaired. If not, at least make the process more transparent by publicly reporting its processes and providing information on how much is being withheld (e.g., the number of vulnerabilities processed, in process, currently withheld, and not released after 30 days, along with statistics on processing time). It should also ensure that the VEP voting membership and chair have a preference for fixing vulnerabilities instead of exploiting them.
- **Identify critical OSS projects/foundations and support them to improve their security, through funds or other contributions.** One example is the Alpha-Omega project, a technical initiative of the OpenSSF. Alpha-Omega provides a path for industry to come together to catalyze critical OSS projects and foundations as they improve their OSS security in systemic and durable ways. It does this by funding security staffing, ecosystem-wide improvements, project audits, and security tooling. We would also like to see the OS3I supporting a government funding parallel to Alpha-Omega, similar to Germany's Sovereign Tech Fund or the differently focused Open Technology Fund.
- **Provide clear Information on existing R&D tax credits for OSS.** The US research and development (R&D) tax credit was created to reward innovation. However, many small organizations don't know about it or understand how to apply it, including how it could apply when improving the security of OSS. Further guidance from the IRS on the availability of tax or payroll credits could encourage increased contribution (either through project funding or direct engineering efforts) to open source software development and security enhancements.
- **Include OSS Contributions as an element of Liability Safe Harbors.** Another approach the government should consider is tying participation in or funding of open source communities (with the specific goal of enhancing secure development and vulnerability response) to immunity from liability related to use of OSS components. If the government is interested in establishing liability for software developers in the wake of vulnerabilities to deter risky behaviors, it is incumbent upon it to also establish liability shields to incentivize desirable behaviors and outcomes. Establishing liability shields for private sector entities as a behavioral incentive is something the government has done before with immensely positive results. Examples include the Cybersecurity Information Sharing Act of 2015 (liability protections for sharing threat indicators or defensive measures), the SAFETY Act (liability protections in the wake of a terrorist attack for products/services that obtain a certification beforehand), and the PREP Act (liability protections for entities and individuals involved in the development or administration of medical countermeasures during a public health emergency)
- **Use government procurement processes to incentivize industry contribution to OSS.** The US government has a long history of creating de facto standards through its formidable purchasing power. Incentivizing industry to partner financially with projects they have levered in winning contracts creates a virtuous cycle of investment in a more

robust and secure software economy without investment of tax dollars. We would encourage the US government to use the power of public procurement, much as they have done so to encourage minority businesses as an example, by creating a special consideration for vendors providing secure software solutions who can demonstrate that, when their solutions stack includes open source software components, they are contributing back to those projects either financially or through direct engineering efforts. Contributions should include those that improve the security and resiliency of the software over time. There are numerous methods to qualify and quantify contributions (public repositories, data from R&D tax credits and so forth).

- Our recommendation would be to form an expert group to consider what approach to take to describe such a system, potentially creating a neutral third party database to maintain a contribution registry. Such a group would include subject matter experts representative of the open source ecosystem with an eye to assisting the federal government; industry, public benefit foundations (including OpenSSF), academia, and other stakeholders with an interest in improving the rate of contribution to the open source projects the US government has come to rely on via their suppliers.

Reduce entire classes of vulnerabilities at scale by securing software repositories

In many cases OSS is obtained as a compiled package from a package repository such as PyPI for Python, RubyGems for Ruby, Maven Central for Java, NuGet for .NET, and so forth. Other repositories for pre-compiled results exist for system packages, container images, and virtual machine images. These repositories are a focal point for all manner of supply chain attacks where attackers can leverage techniques such as typosquatting and dependency confusion.

To tackle the security of OSS supply chains at scale, a key focus must be on securing these repositories. Because each repository is managed by different entities, foundations, or corporations, and handles different types of software distributed in different ways with different installation mechanisms, approaches will need to be tailored for each. Funding and/or experts' time is a key limitation. Many are inadequately funded; even those with commercial backing are often little monetized and are costs to their commercial backers.

We believe the US government should work with these repositories, including either direct funding or using their experts to help repositories improve their security. We recommend that the government work through the [OpenSSF Working Group on Securing Software Repositories](#) (at least in part) to prevent duplicative effort and enable sharing of approaches.

Recommended action:

- **Encourage MFA.** Encourage all package repositories to support various multi-factor authentication (MFA) mechanisms and move towards requiring MFA to update packages. Stolen passwords by themselves should not quickly lead to subverted

packages. Many open source package managers are operated by non-profit foundations, and have limited resources to fund security improvements. There are known high-impact comparatively low-effort security capabilities for these ecosystems, like Two Factor Authentication (2FA) using one-time passcodes, hardware keys you plug into your device, or features built into your device like Mac TouchID or Windows Hello. Per the [Package Manager Security Landscape Survey](#) done by the OpenSSF Securing Repos Working Group, several key ecosystems do not support any MFA2FA like Java's Maven Central or Gradle, or only support one-time passcodes such as RubyGems. The US Government could fund this work, similar to how the Open Technology Fund funded this work for Python's PyPI in 2019.

- **Audit key OSS repositories.** Fund reputable organizations to do security audits for popular package repositories not controlled by a commercial organization (e.g., PyPI) and fix any significant problems found. For repositories controlled by commercial organizations encourage the repository to conduct a security audit (e.g., see [OSTIF's process](#)). This should include detailed threat modeling exercises to identify potential capabilities to prevent and detect problems preventive, detective capabilities to put in place to protect the repository. Also examine whether full time security staff are required to monitor the attack surface of the repository and react accordingly.
- **Simplify digital signing.** Develop and contribute code, or fund mechanisms for easy digital signing of packages and recording provenance in package managers (e.g., see [npm's package provenance work](#)), building on sigstore.
- **Enable malicious package detection.** Encourage package repositories to check for malicious indicators and provide security testing of code before allowing contribution of a package to prevent the contribution of trivial malicious software at its source.
- **Develop, provide and/or promote best practices for packages.** Create this documentation for package management and package creation across popular repositories and languages. This documentation would contain practices such as constraining script execution on installation by default (in those package managers that allow such execution). Encourage repositories to adopt these best practices as project policy.
- **Ensure package managers are secure by default.** Develop capabilities and best practice guidance for consumers of open source software via package repositories to ensure package managers are secure by default (e.g., ensuring that by default installation scripts are disabled (possibly allowing known-safe actions)).
- **Implement secure builds.** Develop capabilities to build software artifacts in secure facilities providing artifacts along with associated metadata. Embedding these capabilities within repositories to ensure any software artifact offered by the repository was securely built and associated with the expected artifact. This would remove multiple attack vectors where attackers replace or infiltrate the supply chain between the developer of the source and the recipient of the software package. Encourage high security industries or government to leverage software from such repositories.
- **Ensure the creation of provenance data to verifiably link a built package back to its source code and build instructions.** There are often no verifiable links from a package back to its source code and build instructions, instead you have to trust that a

package contains what it claims. Several groups in the OpenSSF supported the release of a new security capability in JavaScript's npm package manager called package provenance, which verifiably links a built package back to its source code and build instructions so a human or machine can vet the package's functionality. The US Government could help other repositories adopt package provenance, and encourage OSS maintainers to publish their packages with provenance. The OpenSSF Securing Repositories Working Group has documented more information at [Build Provenance for All Package Registries](#).

- **Implement independent verification for [reproducible builds](#) and [semantically equivalent builds](#) in repositories.** This enables awareness of the heightened risk when using packages that fail such checks and avoiding such packages. Avoiding higher risk packages counters attacks where the built package does not match its putative sources but the widely-used build software is trustworthy. Implement mechanisms for detection of simple malicious packages and suspicious package upload behavior in repositories, to counter naive attackers. Repositories could still allow others to submit a compile package and then attempt to verify that it reproduces (or is semantically equivalent).

In addition:

- **Develop a security metrics dashboard.** This would be a single place to learn about OSS's security status. The OpenSSF has begun work on such a dashboard, but doesn't have the resources to complete it yet. It would be helpful for the government to financially support this work.

Education

It is rare to find a software developer, however educated or trained, who receives formal training in writing secure software. A modest amount of training — 10 hours at the very least, 40-50 hours ideally — could make a huge difference in developer performance. There exist such training modules available for free, such as the OpenSSF Secure Software Fundamentals. Complicating the lack of trained developers is the ever-growing [shortage](#) of trained cyber security professionals that can assist developers. In addition, continuous education is important as new attacks surface.

Recommended action:

- **Ensure that software developer education includes security.** Ensure developing secure software is part of software development education in universities, community colleges, trade schools, and bootcamps. This must include knowing security design principles (including those identified by Saltzer and Schroeder), the most common vulnerabilities and how to counter/mitigate them, common types of tools/analysis for detecting vulnerabilities, and how to handle vulnerability reporting. Since on average software is 80% OSS, they should also know the basics of OSS, including how OSS is developed, how to evaluate it, and how to contribute to OSS projects effectively.
- **Engage with post-secondary education, current software engineers, and primary and secondary students.** Influence curriculum to provide application security and

secure development best practices for learners of all levels as referenced in the [National Cyber Workforce and Education Strategy](#).

- **Mobilize historically underserved communities.** Include education, outreach, and mentorship to expand the diversity of the profession's pipeline and continue to develop development and cyber security technical expertise as referenced in the [National Cyber Workforce and Education Strategy](#).
- **Work with organizations embedded in colleges and universities to promulgate training materials and speakers, e.g., ACM and IEEE chapters.** This is a way to reach students with supplemental material.
- **Ensure that managers of software developers know how to develop secure software enough so they can hire developers with these skills.** This will equip managers to make better hiring choices and help them understand the importance of prioritizing security activities in developer backlogs to remediate found vulnerabilities.
- **Educate OSS consumers/"end users" so they have tooling and signals to understand the security qualities of the software they are ingesting into their enterprises.** Examples include Software Bill of Materials (SBOM), Vulnerability Exploitability eXchange (VEX), Supply-chain Levels for Software Artifacts (SLSA) levels of Continuous Integration/Continuous Delivery (CI/CD) systems, pedigree, provenance, and digital authorship/signing of software artifacts. Expand messaging and communication as security vulnerabilities are discovered, patched, and are available to downstream consumers to increase the speed at which critical software updates are deployed to end-user systems.
- **Promote digital badges and certificates for those developers that complete security training and certification courses.** These are publicly-visible methods of demonstrating security expertise.

Incident response

Incident response was not included as a potential area of focus in the RFI, however we believe it is critical. Prevention is vital, however detection and response is also vital.

The OpenSSF works to help the OSS ecosystem effectively intake, evaluate, address, and disclose security vulnerabilities within their code bases and dependencies. By providing documentation, training, mentorship, and tooling, the OpenSSF Foundation provides numerous methods through which developers, security researchers, and OSS consumers can positively interact within a coordinated vulnerability disclosure (CVD) process. The OpenSSF Vulnerability Disclosures Working Group seeks to help improve the overall security of the open source software ecosystem by helping develop and advocate well-managed vulnerability reporting and communication.

Recommended action:

- **Fund and support the creation of an OSS-SIRT (Open Source Software - Security Incident Response Team)** to work with maintainers, security researchers, and OSS

consumers to ensure effective alignment and communication around the disclosure of OSS vulnerabilities.

- **Provide grants** to enable critical projects the ability to establish or augment existing security response teams, acquire security auditing, tooling, and other security initiatives the project requires.
- **Implement tabletop exercises** to include countering vulnerabilities in OSS components within critical infrastructure sectors, including funding to help enable foundations to attend, and include the Cyber Safety Review Board (CSRB) to analyze simulated results. Work with the CISA Joint Cyber Defense Collaborative (JCDC) open source software (OSS) core planning team (CPT).
- **Modernize Vulnerability Identifiers and Vulnerability Databases:** Increase collaboration and modernization of the CVE Program and the National Vulnerability Database (NVD) to become more agile and compatible with other processes (such as Open Source Vulnerabilities (OSV), VEX, and others). Ensure all future CVEs include at least one package URL (purl) or software ID (cpe - Common Product Enumerator) that provides a machine-readable identifier for affected software without requiring a previous centralized assignment from NIST. Support standards such as VEX for compatible sharing of affectedness data throughout the open source supply chain, originating with upstream developers, flowing through software suppliers and platforms, and ultimately consumable by downstream end-users and consumers.
- **Implement standardized vulnerability reporting and disclosure** in open source projects through tools such as VEX as well as working with downstream security scanners to effectively understand and process VEX statements.
- **Provide training and best practices guidance** to open source communities through industry Coordinated Vulnerability Disclosure Guides for Maintainers, Security Researchers, and OSS Consumers.

R&D / Innovation

The Federal Government has a role to play in fostering research and development of memory safe languages and frameworks. Through the use of research grants, the Federal Government can lead moon-shot ideas to fruition.

Recommended action:

- **Fund R&D in improving security in the use of large language models (LLMs) within systems.** E.g., on how to effectively counteract malicious actors intent on seeding malware or backdoors into production code through training data. The newly released [OWASP Large Language Model Top 10](#) is a start, but more needs to be done.
- **Fund R&D to increase the likelihood that generated code (e.g., via LLMs) is secure by default.** Today it is likely to be insecure. Fund or encourage industry to roll out improvements to their AI augmented code authoring tools, such as GitHub's CoPilot, so that it will offer secure by default code and snippets.

- **Fund R&D in OSS tools for formal verification of code.** This should build on existing systems. For example, developing improved tools for Rust as an additional incentive to adopt .

Foster the adoption of memory safe programming languages, techniques, and tools

Adopting memory safe languages in new and existing code can eliminate most memory safety vulnerabilities. However, it is not possible when there is no appropriate tooling for the target architecture. Also, refactoring large existing code bases into a new language is typically non-trivial. It can introduce new bugs and vulnerabilities, and in some cases it is difficult to redeploy new object code (e.g., in OT/ICS devices). We encourage new development to occur in memory-safe languages where practical. Organizations should take a risk-based approach for existing code, focusing their refactoring efforts where it will be the most beneficial; it would be too costly and risky to try to rewrite *all* software currently in memory-unsafe languages. Focusing on rewriting the riskiest components of the most important software is a practical approach for existing software.

Memory safe languages are *not* a panacea or silver bullet. There are other kinds of vulnerabilities, sometimes the protections must be disabled, and programs will often use memory-unsafe languages.

Recommended action:

- **Create a funded “memory safe program”** to encourage development of new code in memory-safe languages, aid in transitioning the most important code in memory-unsafe languages to memory safe languages (often as specific portions of the software), work to identify and remove roadblocks in the use of memory safe languages (e.g., improve education and tooling for memory safe languages), and encourage deployment of mechanisms to reduce risks when memory unsafe languages are used. This would apply to both code developed for the US government and code used by the US government, industry, and civil society.

Below is a sample roadmap for a future program:

- Support federal government agencies with funds to review and write or rewrite their most critical code (including OSS they use) in a memory-safe(r) language and frameworks, where that is not already the case and to support writing new code in memory safe languages. Where the US government modifies or translates existing OSS, it should release it back upstream for further public review and work.
- Aid [Prossimo](#), which is working to move some of the Internet's security-sensitive software infrastructure to memory safe code in Rust. At the least we would recommend funding the cryptographic library Rustls and its dependencies,

potentially including FIPS 140 evaluation for it, as cryptographic libraries underpin security everywhere.

- Aid [GCC RS](#), an implementation of Rust on GCC. This would enable the use of Rust on architectures Low Level Virtual Machine (LLVM) does not support, which would eliminate an objection for accepting Rust code. It would also enable countering advanced supply chain attacks on compilers themselves.
- Fund LLVM and GCC maintainers to ensure that implementations of memory safe languages such as Rust are available on common IoT and Operational Technology (OT) platforms other than x86-64 and Advanced RISC Machines (ARM).
- Work with tool suppliers (e.g., Static Application Security Testing (SAST) suppliers) to add and improve support for common systems-level memory-safe languages.
- Identify the riskiest areas where memory unsafe languages are used in the federal government and critical infrastructure and develop plans to encourage replacement of the riskiest portions. Include sector coordinating councils and potential agency Open Source Program Offices (OSPOs) in this process.
- Hold a Memory Safety workshop gathering industry, governments, and academia to collaboratively propose the increase the use of memory-safe languages and memory-safe techniques and tools in existing code, including discussions on how to do this most effectively. The OpenSSF can convene and facilitate such workshops.
- Help develop a formal specification of the Rust programming language, building on existing work, to make Rust more adoptable.
- Fund R&D to fix or reduce memory safety and other undefined behavior issues in C and C++, through to acceptance in widely-used compilers, similar to how the NSA invested in SELinux. For example, specifying in source code the number of elements in an arrays that can be checked at compile-time or runtime.