Recommendations from the Workshop on Open-source Software Security Initiative

Angelos D. Keromytis
Georgia Institute of Technology
angelos@gatech.edu

September 2022

Executive Summary

The importance of open-source software (OSS) to the economic well-being and security of the United States (and most of the world) cannot be overstated: OSS components are in pervasive use in commercial products, government systems, and military platforms. The open nature of OSS enables the democratization of software development, rapid evolution, de-duplication of effort on an unprecedented scale, and broad transparency. However, the OSS's decentralized organization, diffuse structure, and large scale make it infeasible to specify or enforce minimum standards for tools and development practices. Combined with the large volume of already-written (legacy) OSS code, this poses unique challenges when it comes to security.

To improve the security of the open-source software ecosystem, a virtual workshop took place on 24-25 August 2022, under the auspices of the Office of Management and the Budget (OMB), the National Science Foundation (NSF), and the National Institute for Standards and Technology (NIST). The goal was to bring together stakeholders from the open-source software (OSS) community, the private sector, academia, and the U.S. Government, in support of the White House-led U.S. Open-source Security Initiative. The workshop specifically focused on recommendations for making progress in the following three topic areas:

- Memory-Safe Programming Languages (focusing on ways to increase adoption in OSS)
- Software Dependency Management
- Behavioral & Economic Incentives to Secure the Open-source Software Ecosystem

This report documents the discussion, findings, and recommendations of the workshop. The document draws upon the notes/writings of workshop participants and scribes; any errors or omissions are those of the author.

The workshop and the author of this report were supported by NSF Grant 2232616. *Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.*

Table of Contents

A. Introduction

Open-source software (OSS) is in pervasive use in commercial products, government systems, and military platforms. Although many OSS projects originally started modestly, as side activities of one or a small group of developers, many have grown to encompass the work of a global community of developers who are dedicated to generating, maintaining, and improving widely used software components. The open nature of OSS enables the democratization of software development, which yields rapid evolution, de-duplication of effort on an unprecedented scale, and broad transparency. However, its large scale, decentralized organization, and diffuse structure make it impractical to develop and specify standard tools and processes. Each project independently chooses tools and technologies that meet the project's goals, and its developers' preferences and habits (as they relate to software development). Combined with the large volume of already-written (legacy) OSS code, this poses unique set of challenges when it comes to security. Simply put, absent an external mandate to commit resources (*e.g.*, developers' time) toward improved security outcomes, it is left to individual projects, teams, and developers to navigate and balance core project interests. These stakeholders choose whether to allocate limited resources towards choosing or changing a programming language (including migration to more modern ones), improving or enforcing software integrity across the supply chain (upstream and downstream), screening code contributions, evaluating the use of new analysis tools, and the adoption of deliberate development practices that can lead to higher assurance. Given the broad societal dependency on OSS, it is in the interest of the United States to identify opportunities for improving on this state of affairs, while simultaneously taking care to avoid negatively impacting the unquestionable benefits derived from the OSS model and its products.

Under the auspices of the Office of Management and the Budget (OMB), the National Science Foundation (NSF), and the National Institute for Standards and Technology (NIST), 12 individuals from academia, industry, the open-source software community, and the U.S. Government were chosen to form a Steering Committee (SC – see Appendix A for the associated biography sketches) and met on 20 May 2022 to discuss four main themes related to the security of open-source software: (1) Developers' Perceptions of Trust & Safety, (2) Memory-Safe Programming Languages, (3) Dependency Management, and (4) Behavioral and Economic Incentives to Secure Open-Source Software. These themes were selected by NSF and NIST staff based on a qualitative analysis on the position statements submitted by each Steering Committee member.

For each theme, the structured discussion focused on answering the following questions[1]:

1. What is the problem? (Define)
2. What is it we don't yet understand?
3. Where are the boundaries of the problem? Are there any constants that can't be changed?
4. Who are the key stakeholders to get involved? Sectors or specific names

To seek deeper discussion in these topics, it was decided to hold a virtual workshop on August 24-25, 2022 as a follow-on to the SC meeting with a broader set of participants (see Appendix B for the associated biographical sketches). The workshop's goal was to bring together stakeholders

[1] Based on the Phoenix Checklist creativity instrument and adapted from Thinkertoys: A Handbook of Creative-Thinking Techniques, Michael Michalko. Ten Speed Press; 2nd edition (December 1, 2010)

from the open-source software community, the private sector, academia, and the U.S. Government, to identify ways to improve the security of the open-source software ecosystem. This workshop was intended to benefit the United States by identifying ways to investing in the shared open-source software infrastructure that the public and private sectors both rely on and addressing challenges spanning the global software community. Furthermore, the workshop was in direct support of the White House-led U.S. Open Source Security Initiative.

The workshop consisted of a combination of invited talks, panels, and breakout sessions. (The detailed meeting agenda can be found on the public workshop website.)

The workshop specifically focused on making recommendations for progress in the following three topic areas, combining the Themes 1 and 3 from the Steering Committee meeting into one:

- Memory-Safe Programming Languages (focusing on ways to increase adoption in OSS)
- Software Dependency Management
- Behavioral & Economic Incentives to Secure the Open-source Software Ecosystem

The attendees were told that the scope of the recommendations generated by the workshop may cover U.S. Government Research and Development (R&D) investment (including potentially sponsoring a Grand Challenge), acquisition practices, policy and legal issues, and other mechanisms through which OSS security may be improved for all.

This document contains a report of the discussion and recommendations generated through the workshop.

B. Discussion and Recommendations

The discussions – both at the Steering Committee meeting and in the subsequent workshop – indicate that the topics we focused on are strongly interconnected. Nonetheless, we chose to report on each of these topics separately to better reflect the context of the discussion and to impose some structure that aids the comprehensibility of the recommendations.

B.1. Memory Safety

The main participants in the Memory Safety breakout group were Nikhil Swamy (moderator), David Brumley, Matthias Payer (EPFL), Per Larsen, Lin Clark, David Tarditi, Nicholas Matsakis, Deian Stefan (scribe), Lars Bergstrom, Josh Aas, and Luke Wagner. Athanasios Moschos served as an additional scribe. USG representatives from various departments and agencies (OMB, NSF, NIST, the Department of Homeland Security (DHS), the Office of Naval Research (ONR), and the Office of the National Cyber Director (ONCD) participated as observers.

B.1.1. Context

The software vulnerability landscape has not significantly changed in the last decade despite various innovations in exploit mitigations (*e.g.*, Address Space Layout Randomization (ASLR), Control Flow Integrity (CFI)) and software testing techniques (*e.g.*, fuzzing and sanitization). Several recent reports indicate that roughly 70% of all software vulnerabilities continue to be memory safety problems that arise from the use of memory unsafe languages such as C or C++. One of the key questions debated by the Steering Committee was how to (gradually) encourage the transition software developers to use memory safe languages.

The committee discussed several problems that have impeded this transition. Getting developers enthusiastic to learn and write in a new language while meeting their performance expectations is hard. Also, there is a large amount of legacy code and converting it will be a manual, cumbersome process that will take years.

To make significant progress on this theme, we need a better understanding of various aspects of the problem area and its potential solutions. This includes identifying a clear migration path for projects, tactical community interactions, sustainable funding, and automation. While progress on this theme will mitigate memory corruption vulnerabilities, there is still the continued need for secure coding practices to prevent other classes of bugs (*e.g.*, design flaws, information disclosure, or DoS).

The Steering Committee identified the need to change certain aspects of the economics and mechanics of software development to be successful. We need to incentivize developers to write secure code and make security a part of their professional training and university/college course curriculums. The transition to memory safe languages is several years away, but we can speed it up by increasing investments in ease-of-use, interoperability, and automation.

B.1.2. Workshop Discussion and Recommendations

According to reports mentioned by the workshop participants, 60%-70% of all vulnerabilities are related to memory safety issues. The group thought that it is important to calculate the total societal cost from these bugs, as well as the cost to national security, as a way of motivating developers and organizations to voluntarily adopt memory-safe languages (MSLs). This should be accompanied by a concerted effort to shift the narrative around MSLs to encourage their use, at least in new projects where it is easier to do so. Tools that lower the barrier to entry should be developed for programmers that are not versed in MSLs and their ecosystems. These tools may include code translation, analysis, Integrated Development Environments (IDEs) optimized to support experienced programmers learning/developing in a new language, and tools for making it easy to integrate safe and unsafe languages that encourage incremental conversion. While some of this work is being done in current Research & Development (R&D) programs sponsored by the USG (*e.g.*, NSF's Secure and Trustworthy Cyberspace), what is needed is a renewed focus on practical tools along with a targeted outreach to the OSS community.

On the topic of MSL adoption, the workshop identified two different timelines, a near-term and a longer-term one.

Near-term recommendations: In the near term (next two years), we should identify software (or portions of software, *e.g.*, input parsers) that are in an elevated risk for exploitable vulnerabilities, and focus on porting them to MSLs. More generally, research is needed to characterize the categories of projects that will benefit from a transition to MSL, and to offer/develop solutions for each category. Providing detailed examples of such transitions as guides to other developers may help demystify the process for those that are amenable but worried about the amount of effort involved.

The workshop consensus was that the Rust programming language, despite its initial learning curve, is particularly well-suited for safe systems-level development, even where performance requirements are important. Where performance requirements are less stringent, languages with garbage collection (GC), such as Java, Python, or Go, may be easier/simpler to use. Toolchains can encourage wider adoption of MSLs for programs, and thus, funding should be driven to the development of tools for MSLs. High performance MSLs like Rust should be well funded and enhanced with more tooling (*e.g.*, static analysis tools, C2RUST converters, compiler plug-ins that facilitate translation to Rust/WASM), to make its ecosystem more robust.

The neediest ("scariest") areas should be identified and prioritized for adoption of MSLs. A specific objective would be to focus on at least partial rewrite in MSLs of frontline code such as parsers and codecs, starting from widely used parser libraries. Not everything written in C/C++ can be rewritten, thus priority should be given to things that can actually be effectively converted. The DARPA SafeDocs program could serve as a model for developing relevant technology. As a cautionary note, care must be taken to avoid OSS project forking that would lead to unnecessary variants. Care must also be taken to understand and control the effects of stitching together different code segments (*e.g.*, MSL with non-MSL code segments) and to enable communication between them, such that we do not introduce errors in the MSL portions of the code. It is a matter

for research to determine how this can be done in a safe way that will not compromise the security of the host system, even if the non-MSL part of the code is exploited.

Any new cryptographic library (*e.g.*, *zk-snarks*, *pquantum*) should be implemented in a language like Rust, especially the ones submitted to NIST as part of calls for standards. This seems like a particularly "low-hanging fruit" type of task.

Ubiquitous embedded IoT systems should be a particular target for use of MSLs. Especially as it pertains to their use in critical infrastructure (*e.g.,* energy, utilities), some combination of regulatory action and insurance requirements/premiums may nudge the market in the right direction.

A complement to the use of MSLs is the adoption of sandboxing. Monolithic software should be broken into components where transitions between components are checked according to a compartmentalization policy. This requires less effort and is easier to implement (no need to learn a new language) but is generally harder to maintain and only serves as a partial solution as the original bugs still exist in the source code. An incremental, migration-focused approach can be followed, where initially we compartmentalize the code and apply sandboxing solutions. Then we gradually move more and more of the compartmentalized code to MSLs.

Longer-term recommendations: In the long run, what is needed is the wider adoption/learning of MSL, typically for new projects (even realizing that not all new projects can be written in an MSL).

Not everyone or every project or organization will be able to adopt and or transition to MSL because there are a large legacy code base and interoperability between components written in different languages is hard, and when developers already have a programming language of choice, it is hard to convince them to change. Universities can help with the latter by refocusing programming curricula to use MSLs and to educate their students about the cost of using MSL *vs.* non-MSL solutions. Research is needed in figuring out how to measure the value of converting a program in an MSL: we need to articulate that to software developers who are not security specialists and convince them about the importance of adopting MSL for software development; these evaluations should be provided from security-experts to non-security-experts. Importantly, modern MSLs also offer inherent support for safe, high-performance concurrent programming, which is key in the multi-core era. NSF support for such curriculum refocus may be appropriate. Furthermore, requirements for evidence-based security (as part of USG procurement rules, insurance mandates, *etc.*) would ease the adoption of MSLs in general, since auditing/proving security is easier than in the case of C/C++ code.

Can an economics argument be made for the use of MSLs? The community has many painful stories of failure on migration to share. However, there are also stories of success that should be shared. For example, anecdotally, Java/Rust teams appear to be smaller (hence likely cheaper?) than teams writing the same code in C/C++. Research is needed in measuring engineering costs over the lifetime of a project, for different languages, the costs and value of migration/transition, and the categories of projects that will benefit from transition with solutions offered per category.

This research could be complemented with case studies, especially of before/after transition to the use of an MSL.

Transparency may provide the data needed and empower user choice. As a minor point, the Software Bill of Materials (SBOM) should explicitly include memory safety characteristics of the software (including its individual components), rather than relying on inferring it from the use of language(s) used.

MSLs provide a base safe level of assurance. While bugs will still exist, exploitability becomes dependent on the environment and location of the bug, compared to memory safety vulnerabilities that provide general and broad exploit primitives. There was therefore consensus that once we achieve memory safety, adversaries will have to resort to much harder per-use-case exploits that are much more costly to find and exploit. We have already been witnessing many of the apps that have more than one million users getting rid of their C code due to the high cost inflicted by memory vulnerabilities.

In terms of measuring success, lists of common vulnerabilities and exposures (CVEs) are a solution, albeit a weak one, as there is bias from vendors when it comes to disclosing certain information about the nature of the vulnerabilities. For example, vendors may not be reporting CVEs for bugs found internally, so CVEs cannot serve as a measurement of safety. Thus, incentives should be provided to vendors for higher transparency.

B.2. Software Dependency Management

The main participants in the Software Dependency Management breakout group were David Wheeler, Sumana Harihareswara, Deirdre Connolly, Anil Madhavapeddy, William Bartholomew, Joshua Lock, Luke Hinds, Dustin Ingram, Rhys Arkins, Mel Chua, and Justin Hutchings. Athanasios Avgerinos served as the group scribe. Daniela Oliveira served as the moderator. USG representatives from various departments and agencies (OMB, NSF, NIST, DHS, ONR, and ONCD) participated as observers.

B.2.1. Context

Software reuse is on the rise. Instead of writing code for a needed functionality themselves, developers are increasingly relying on external libraries to fill that gap. This practice enables modularity, increases code reuse, and reduces software development cost. Similarly, if a dependency is well developed and well maintained then the overall security of the software will benefit. One example are cryptographic libraries that should be reused, and developers being encouraged to never implement their own cryptographic code. The downside is that these dependencies become liabilities that need to be carefully managed. Any code that is included becomes part of the trusted computing base but remains under external control.

Key challenges of leveraging external dependencies are *(a)* tracking all dependencies and their transitive hull of dependencies, *(b)* keeping the dependencies up to date, which incurs software development costs due to changed functionalities but is required to the bug patches, and *(c)* detecting if any used feature has changed or requires an update. While bug fixes should be applied

immediately, a developer may hold off on updates with feature changes as they will require changes to their code.

Software dependencies are evolving, and different languages and runtime environments have slightly different concerns and challenges. We must better understand the scope of the problem, such as how deep dependency chains are or how fast software changes. Alongside, we must define metrics on the security of dependencies and develop technological solutions that empower developers to keep track of their externally included code.

The Steering Committee's view was that both research into solutions and metrics along with development of better tooling will be necessary to address these challenges. Dependency management is an important and challenging problem that needs to be solved by integrating the solutions gradually into the developer workflow, enabling them to react to challenges whenever dependencies are updated.

B.2.2. Workshop Discussion and Recommendations

Software is mostly reused components, and most of those are open-source software. Thus, vulnerabilities in reused components are a common cause of vulnerabilities in larger systems. The main two security issues relating to software dependency management (SDM) are *(a)* identification of dependencies on accidentally vulnerable software (*e.g.*, CVE discovered in OSS library incorporated in project), and *(b)* malicious packages (trojan). These security issues require different threat modeling; however, several workshop participants asserted that 99% of risks in OSS currently comes from unpatched CVEs. Either way, one critical missing component is software provenance tracking – a key issue, even for otherwise safe languages (*e.g.*, Rust).

SBOM can be a key component of any solution, but the associated tooling for creating, manipulating, and generally integrating with development and management processes and workflows is currently viewed as immature. Anecdotally, workshop attendees expressed the opinion that the majority of SBOMs are "worthless" or "security theater". What is needed is canonicalization of software names/references, and tools for managing/maintaining and checking the dependencies in an SBOM. At a minimum, software manifests should include transitive dependencies. Ideally, the SBOM functionality should be integrated into the build tools by default; a targeted R&D effort could yield great results in this space.

Software package managers vary in terms of trust properties provided (*e.g.*, curated *vs.* non-curated). However, successful open-source ecosystems appear to be permissive/open. Rather than try to change that, we should strive to provide the tools within the ecosystems to enable informed decisions to be made by users and downstream maintainers. This can include basic information about the developers and the software versions, but also other metadata of interest, such as audit results. To complement this, the USG could facilitate the creation of a marketplace for package/project audits, trying to leverage expertise and share the results. Another R&D initiative would focus on sponsoring security improvements in package managers.

However, the workshop participants noted that even if we track dependencies, nothing tracks the metadata on how projects are maintained. For example, if control of an OSS project changes, there

are new trust decisions to be made by its downstream users. We need mechanisms for tracking such changes continuously and including them in some registry.

The workshop participants identified of the need for higher quality security data, such as machine-readable CVE data. The perception of the OSS community is that this is easily within the power of the USG to do via MITRE and will likely have a big impact in any other tooling needed for SDM.

Furthermore, we do not understand the processes and workflows of maintainers and projects outside of GitHub or centralized platforms. Sponsoring the mapping of the open-source ecosystem dependencies on a continuous basis would improve transparency, help identify existing and emerging risks, and enable further targeted work. For example, such a map can be used to identify critical projects even as they emerge. A potential starting point is the OpenSSF's list of critical projects.

Most dependency management systems can only reason about well-structured dependencies. There is not a clear path to identifying dependencies introduced via forking by a vendor, copy/pasting, *etc*. outside of these well-structured definitions. Tooling for such "fuzzy matching" of software is sorely needed and needs to be integrated with SBOM tooling.

Some (perhaps much) of what was discussed in this breakout group could probably be achieved (and certainly positively influenced) by adjusting USG procurement rules.

B.3. Behavioral & Economic Incentives

The main participants in the Behavioral & Economic Incentives breakout group were Laurie Williams (moderator), Abhishek Arya, Alex Gaynor, Uma Karmarkar, Yasemin Acar (co-scribe), Deborah Shands (co-scribe), Marshall Van Alstyne, Anne Bertucio, Shane Miller, Greg Kroah-Hartman, Bob Callaway, and Georgia Bullen. USG representatives from various departments and agencies (OMB, NSF, NIST, DHS, ONR, and ONCD/EOP) participated as observers.

B.3.1. Context

A key question posed by the Steering Committee related to how we can incentivize cyber-resiliency in the teams of the volunteers that build OSS. Resiliency includes both creating more secure software and creating robust systems for maintaining that security throughout the software lifecycle.

The Linux Foundation reports that "A key challenge is that OSS contributors are volunteers, and that efforts focused on dramatically increasing the time current contributors spend on security are unlikely to be welcome." Studies also show that most OSS developers have primarily non-monetary motivations, thus purely monetary incentives (*e.g.*, paying for better maintenance) might be insufficient.

However, we do not currently have a good understanding of what incentive structures would work to drive faster forward progress. Thus, the consensus is that the U.S. needs to fund and conduct more research into the economics of cyber-secure OSS development in at least three areas:

1. **Software Development Lifecycle.** What incentive structures would change volunteer OSS developers to incorporate better software security practices? And how do we provide this information to developers? And during the software lifecycle, what incentive models encourage active and speedy remediations when issues are discovered? What if there is only one primary developer, and they leave the project? Are incentives purely contributor based, or are there other incentive models such as matching security experts to OSS projects?

2. **Using Paid Professional Development/Assistance Wisely.** How can we leverage the limited amounts of government, industry, and professional assistance to maximize our overall security posture? How do we assess the security of OSS dependencies? And how do we ensure that we encourage startups rather than large existing businesses? Can we identify the weakest links, shore up security, and then rinse and repeat with the next highest priority? For example, companies often professionally support OSS projects (*e.g.*, through grants, compute resources, or paid on-going development), but cannot support every single project they depend upon. How does the U.S. government, as well as individual companies, get the most "bang for the buck" out of the limited professional funding and development we can put behind OSS?

3. **Informed Choice.** Today OSS consumers (commercial users, governments, and even other OSS projects) do not heavily weigh security when making choices. How do we make security a discerning feature?

These challenges also require that we create metrics that measure whether security is improving or not. We do have some data, such as the total number of vulnerabilities, and the total number of new OSS projects created each year. However, we lack a code index metric that allows us to determine whether we are trending to more or less secure over time.

B.3.2. Workshop Discussion and Recommendations

Software is produced and consumed by humans. Thus, to secure the open-source software ecosystem, it is crucial that the community understands and considers the socio-technical issues involved, such as who the participants (or stakeholders) of this ecosystem are, what challenges they are current facing, and what drives their behaviors.

The group discussion classified the participants of the OSS ecosystem into two main categories: producers and consumers of OSS code. Producers are individual maintainers or developers who contribute (*e.g.,* with code) to the OSS ecosystem. These developers might be non-paid individual volunteers not housed within a well-resourced organization or paid professionals whose contributions to the OSS community are supported by their companies. Developers are generally busy and overwhelmed with their primary tasks of producing new functionalities and lack incentives for acquiring and exhibiting a security mindset while working on their primary tasks.

Organizations that develop OSS are also producers. Interestingly, producers are currently not liable for the (insecure) software they develop or any harm they software can inflict upon their users. When developers use OSS via dependencies in their own code and organizations have products or services that depend on OSS, they operate as consumers or end users of OSS. The main challenges faced by consumers is a lack of *(1)* trust in the code being consumed, *(2)* transparency on their dependencies of third-party software, *(3)* guidance and awareness on how to consume OSS in a trustworthy manner, and (4) situation awareness for when trust is violated, *e.g.*, when a critical vulnerability is found in a software component they consume or depend on.

The key question tackled by the group was: how to create and maintain behavioral and economic incentives for these diverse stakeholders to operate in a way that foster a more secure OSS ecosystem? There are several challenges to answer this question, given that the incentives driving producers and consumers in their different capacities (individuals and organizations) are not necessarily well aligned and might be even at odds with one another. For example, while producers have an incentive to deliver functionality as fast as possible (without necessarily accounting for security - "it's the consumers' responsibility"), consumers would like to trust the components they depend on.

After much discussion among the workshop participants, the main takeaway is that there are more unknowns than knowns for this theme. The reason conjectured is the OSS community attempting to tackle these socio-technical issues solely through a "computing" lens. The problem is multi-disciplinary, and solutions should involve expertise in computer science, software engineering, cyber security, social psychology, and behavioral economics. The recommendation is catalyzing multidisciplinary research on the topic, an endeavor in which the NSF can play a crucial role. The following research & development questions were posed:

Economics

1. What are the incentives driving each stakeholder?
    1.1 How can we leverage/align these incentives among this diverse group to create a more secure OSS ecosystem?

2. How to characterize/measure a project level of exposure to security risks?

3. How to characterize trust (in developers, projects, organization, and dependencies) in the OSS ecosystem?

4. How to support the wide variety of project sizes, especially the very small contributor team on a very important project?

5. To what extent accountability/liability helps in fostering security practices in software development?
    5.1. How to operationalize and implement accountability/liability?

6. How can we measure the Return of Investment (ROI) of implementing secure software development practices in OSS projects?

7. What is/How to measure the economic harm of keeping the status quo?

Software Development Practices

8. Why people decide to adopt or not a secure approach?
    8.1 How to measure this quantitatively?

9. To what extent security audits helps in preventing security risks and vulnerabilities?

10. In which conditions does monetary compensation translates into more secure software development practices, if at all?

11. How to include security training that is appropriate for everyone?
    11.1 How to tailor security training to a developers' profile - seniority level, security experience, project size/type, and organization size/sector?

12. To what extent hiring security specialists for OSS projects translates into more secure software being developed?

13. What type of usable security tooling is needed to streamline secure software development practices?

Team Dynamics

14. How to detect social engineering attacks (*e.g.,* npm event stream)?
    14.1. How to identify attackers?
    14.2. How to identify social engineering tactics of persuasion?

15. How to detect suspicious developers' ascendency in projects?
    15.1. How are people embedded into OSS projects?

16. Is toxicity in team communications associated with less secure software development practices?

Evaluation

There is no consensus on what indicators of success should be.

17. What are the criteria to evaluate success in adopting socio-technical measures for a more secure OSS ecosystem?

Although most of the discussion was an acknowledgement of how much the community does not know, a few recommendations were uncovered. First, OSS maintainers should be financially compensated. There should be requirements to pay into organizations that support community and/or security practices if one gets a government contract for projects that use OSS. Second, there

is a need for tooling to support dependency transparency (e.g., SBOM metadata, including the presence of code in unsafe programming languages), mechanisms to support the reporting of suspicious packages, visibility of to what extent OSS is supporting a given project, and security/vulnerability scores for code making up a software piece. Third, there is also a need for metrics, which will help the community evaluate the success of proposed socio-technical approaches to secure the OSS ecosystem, such as quantification of vulnerabilities and dependencies, mean time to remediation, and criteria for designating a project as critical.

In sum, there is a lack of understanding on how to build an effective human-in-the-loop ecosystem that creates secure software. Given the multidisciplinary aspects of the problem, collaborative research among computer security researchers/experts and social, behavioral and economics (SBE) researchers is warranted. Furthermore, supportive tooling and metrics that provide transparency and situation awareness about the level of risk and exposure of OSS projects is needed and will likely support future research aiming at the socio-technical aspects and challenges in securing the OSS ecosystem.

Appendix A
Steering Committee Members

Abhishek Arya
Principal Engineer and Head of Google Open Source Security Team

Abhishek Arya is a Principal Engineer and head of the Google Open Source Security Team. His team has been a key contributor to various security engineering efforts inside the Open Source Security Foundation (OpenSSF). This includes the Fuzzing Tools (Fuzz-Introspector), Supply Chain Security Framework (SLSA, Sigstore), Security Risk Measurement Platform (Scorecards, AllStar), Vulnerability Management Solution (OSV) and Package Analysis project. Prior to this, he was a founding member of the Google Chrome Security Team and built OSS-Fuzz, a highly scaled and automated fuzzing infrastructure that fuzzes all of Google and Open Source. His team also maintains FuzzBench, a free fuzzer benchmarking service that helps the community rigorously evaluate fuzzing research and make it easier to adopt.

David Brumley
CEO and Co-Founder of ForAllSecure and Full Professor at Carnegie Mellon University

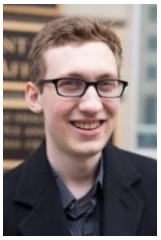
Dr. David Brumley is CEO and co-founder of ForAllSecure and a full professor at Carnegie Mellon University. His accomplishments include winning the DARPA Cyber Grand Challenge, a United States Presidential Early Career Award for Scientists and Engineers (PECASE) from President Obama, a Sloan Foundation award, a Carnegie Science Award, several patents, numerous academic papers, a DEFCON black badge, and mentoring one of the most competitive hacking teams in the world.

Deirdre Connolly
Cryptographic Engineer at the Zcash Foundation

Deirdre Connolly is a cryptographic engineer at the Zcash Foundation. She works on secure implementations of cryptographic software with an eye on privacy applications, misuse-resistance, and an eye on quantum adversaries. She obtained her BS from MIT in 2009.

Alex Gaynor

Deputy Chief Technologist for Security at the Federal Trade Commission



Alex currently serves as Deputy Chief Technologist for Security at the Federal Trade Commission. Prior to that he was at the United States Digital Service. He has previously worked at Alloy, Mozilla, and another stint at the United States Digital Service. Alex has a long history of involvement in the open-source community. He is a core developer of the Python Cryptographic Authority and previously has served as a member of the board of directors of both the Python and Django Software Foundations. Alex lives in Washington, DC and likes delis and bagels.

Royal Hansen

Vice President of Privacy, Safety, and Security at Google

Royal Hansen is Vice President of Privacy, Safety & Security at Google, where he is responsible for driving strategy and implementation in these areas across the company's technical infrastructure and product lines. There, he was responsible for solutions protecting the security and integrity of the company's technology systems and the customer, business, and employee information they processed. Before American Express, Royal served as both the Managing Director, Technology Risk and the Global Head of Application Security, Data Risk and Business Continuity Planning at Goldman Sachs. Royal was also previously at Morgan Stanley and Fidelity Investments, where he managed Enterprise IT Risk, Application Security and Disaster Recovery. Royal began his career as a software developer for Sapient before building a cyber-security practice in the financial services industry at @stake, which was acquired by Symantec. Royal holds a BA in Computer Science from Yale University. He was awarded a Fulbright Fellowship in information sciences and Arabic language study, which he completed at the United Arab Emirates University.

Sumana Harihareswara

Project Manager, Programmer, and Trainer at the Python Software Foundation's Packaging Working Group and Founder of Changeset Consulting

Sumana Harihareswara is a project manager, programmer, and trainer who leads a consultancy working with open source software projects and maintainers. She led the rollout of the next-generation PyPI.org and pip resolver, and has worked on HTTPS Everywhere, Autoconf, Mailman, MediaWiki, and several other open source projects across industry, academia, nonprofits, and volunteer settings. She works with the Secure Systems Lab at New York University on securing the software supply chain in Python and is a member of the Python Software Foundation's Packaging Working Group. She is writing a book on rejuvenating and managing legacy open source projects and teaches workshops in maintainership skills. She earned an Open Source Citizen Award in 2011 and a Google Open Source Peer Bonus in 2018. She lives in New York City and founded Changeset Consulting in 2015.

Angelos Keromytis

Professor, John H. Weitnauer Technology Transition Endowed Chair, and Georgia Research Alliance (GRA) Eminent Scholar at the Georgia Institute of Technology

Dr. Angelos Keromytis is Professor, John H. Weitnauer Technology Transition Endowed Chair, and Georgia Research Alliance (GRA) Eminent Scholar at the Georgia Institute of Technology. He is an ACM and IEEE Fellow, and President of Voreas Laboratories Inc and Aether Argus Inc, two Georgia Tech technology spinoffs. He has served as Program Director with the National Science Foundation and Program Manager at DARPA. His field of research is systems and network security, and applied cryptography.

Mathias Payer

Associate Professor, École Polytechnique Fédérale de Lausanne (EPFL)

Mathias Payer is a security researcher and associate professor at EPFL, leading the HexHive group. His research focuses on protecting applications in the presence of vulnerabilities, with a focus on memory corruption and type violations. He is interested in software security, system security, binary exploitation, effective mitigations, fault isolation/privilege separation, strong sanitization, and software testing (fuzzing) using a combination of binary analysis and compiler-based techniques.

Eric Rescorla

Chief Technology Officer, Firefox at Mozilla

Eric Rescorla is Chief Technology Officer, Firefox at Mozilla, where he is responsible for setting the overall technical strategy for the Firefox browser. He has contributed extensively to many of the core security protocols used in the Internet, including TLS, DTLS, WebRTC, ACME, and QUIC. He was editor of TLS 1.3, which secures over 50% of web sites. To remove barriers to encryption on the web, he co-founded Let's Encrypt, a free and automated certificate authority that now issues more than a million certificates a day, and helped HTTPS grow from around 30% of the web to over 80%. Previously, he served on the California Secretary of State's Top To Bottom Review where he was part of a team that found severe vulnerabilities in multiple electronic voting devices.

Nikhil Swamy

Senior Principal Researcher at Microsoft Research

Nikhil is a Senior Principal Researcher at Microsoft Research (MSR) at its headquarters in Redmond, USA, where he has worked since 2008. His expertise is in programming language design and semantics, formal verification, and software security. He is perhaps best known for his work on F*, a proof-oriented programming language. Verified cryptographic algorithms, communication protocols, blockchain components, and network virtualization software produced in F* are deployed in the Linux kernel, in Windows, in the Microsoft Azure cloud, in the Firefox web browser, and several other industrial software components, improving computer security and reliability for billions of users every day.

David A. Wheeler

Director of Open Source Supply Chain Security at The Linux Foundation

Dr. David A. Wheeler is an expert on open source software (OSS) and on developing secure software. His works on developing secure software include "Secure Programming HOWTO", the Open Source Security Foundation (OpenSSF) Secure Software Development Fundamentals Courses, and "Fully Countering Trusting Trust through Diverse Double-Compiling (DDC)". He also helped develop the 2009 U.S. Department of Defense (DoD) policy on OSS. David A. Wheeler is the Director of Open Source Supply Chain Security at the Linux Foundation and teaches a graduate course in developing secure software at George Mason University (GMU). Dr. Wheeler has a PhD in Information Technology, a Master's in Computer Science, a certificate in Information Security, a certificate in Software Engineering, and a B.S. in Electronics Engineering, all from George Mason University (GMU). He is a Certified Information Systems Security Professional (CISSP) and Senior Member of the Institute of Electrical and Electronics Engineers (IEEE). He lives in Northern Virginia.

Laurie Williams

Distinguished University Professor in the Computer Science Department at North Carolina State University

Laurie Williams is a Distinguished University Professor in the Computer Science Department at North Carolina State University (NCSU). Laurie is a co-director of the NCSU Secure Computing Institute, the NCSU Science of Security Lablet, and the North Carolina Partnership for Cybersecurity Excellence (NC-PaCE). Laurie's research focuses on software security; agile software development practices and processes, particularly continuous deployment; and software reliability, software testing and analysis. Laurie is an IEEE Fellow and an ACM Fellow. Laurie received her Ph.D. in Computer Science from the University of Utah, her MBA from Duke University Fuqua School of Business, and her BS in Industrial Engineering from Lehigh University. She worked for IBM Corporation for nine years in Raleigh, NC and Research Triangle Park, NC before returning to academia.

Appendix B
Workshop Participants

Josh Aas (‘oohse’) co-founded and currently runs Internet Security Research Group (ISRG), the nonprofit entity behind Let’s Encrypt, the world’s largest certificate authority helping to secure more than 290 million websites. He also spearheaded ISRG’s latest projects, one focused on bringing memory-safe code to security-sensitive software, called Prossimo, and Divvi Up, a privacy-respecting metrics service. Josh worked in Mozilla’s platform engineering group for many years, improving the Firefox web browser. He also worked for Mozilla in a senior strategy role, helping to find solutions for some of the Web’s most difficult problems. He has deep expertise in software security and ecosystem dynamics, as well as organizational leadership.

Yasemin Acar an assistant professor at the George Washington University. Her research focus is on human factors in secure development, specifically: investigating how to help software developers implement secure software development practices. Her research has shown that working with developers on these issues can resolve problems before they ever affect end users. Her most recent work aims to support security and trust in open-source software.

Marc Alvidrez is an engineer and project lead at the U.S. Digital Service (USDS), where he has worked on a diverse set of projects that include the launch of **covidtests.gov** and improvements to the national Organ Procurement and Transplantation Network (OPTN). Coming to government after 25 years in industry, he was a member of the first generation of Site Reliability Engineers at Google. Most recently he worked at Loon where he was responsible for integrating, operating, and securing a stratospheric, balloon-based communications platform capable of bringing LTE and Internet service to un(der)served communities around the world.

Rhys Arkins is Vice President, Product Management at Mend. Rhys joined Mend (formerly WhiteSource) in 2019 through the acquisition of his startup, Renovate Bot, an Open Source dependency automation tool. Today Rhys maintains a focus on dependency management and automation to improve developer experience and application security. Rhys was awarded a University Medal for his studies in Information Technology at the University of Queensland, and now resides in Stockholm, Sweden.

Anne Bertucio leads program development in Google's Open Source Programs Office (OSPO). The Program Development Team helps teams at Alphabet develop, contribute to, and release open source software with an eye towards strategy, sustainability, and the spirit of the Open Source Definition. The Program Development Team works across domains, from cloud to data analytics to gaming to security. Security is a special focus for Anne, particularly **open source vulnerability disclosure**. She previously worked on Kubernetes and container security, and authored the paper ***Why Container Security Matters to Your Business***. Before coming to Google, she was a staff member of the OpenStack Foundation (now known as the Open Infrastructure Foundation), where she was part of the inaugural core team of the Kata Containers project and on the OpenStack release management team. Anne has B.A.s in policy and ethics and worked in community and government relations in renewable energy before coming to tech.

William Bartholomew (he/him) is a Principal Security Strategist in the Global Cybersecurity Public Policy team at Microsoft. His public policy advocacy benefits from over a decade of experience in designing, implementing, and operating software supply chains used by tens of thousands of developers. Prior to focusing on public policy, he held engineering and product management roles within Microsoft and GitHub that focused on delivering reliable and secure engineering systems for developers internally as well as for our customers. He brings his relentless focus on reducing friction to standards development, open source, and public- and private-sector working groups globally. When not working, he can be found tinkering with hardware and software, making espresso, and spending time with his family in the United States' Pacific Northwest.

Lars Bergstrom is a Director of Engineering at Google on the Android team, working on their platform programming languages, including Java, C/C++, and Rust. He also serves as Google's Corporate Director to the Rust Foundation. Before Google, he was at Mozilla Research, initially contributing to the Servo browser project and directing the integration of Rust into Firefox and the partner ecosystem. Later, he led Mozilla's AR and VR work, shipping software and building OEM relationships on many different devices. He is currently based out of Chicago, where he lives with his wife and son.

Jon Boyens is the Deputy Chief of the Computer Security Division in the Information Technology Laboratory at the National Institute of Standards and Technology (NIST). His responsibilities include Cybersecurity Research and Development at NIST and Cybersecurity Standards and Guidelines for Federal Agency Security Programs.

Georgia Bullen is the Executive Director at **Simply Secure**, a nonprofit leveraging design as a transformative practice to shift power in the tech ecosystem and change who technology serves. She brings over 15 years of experience in usability, design, technology, policy and research to her work, contributing to the internet health movement on issues such as security, privacy, open source, and equitable access to technology.

Bob Callaway is the technical lead and manager of the supply chain integrity group in Google's Open Source Security Team. He and his team directly contribute to critical secure supply chain projects and drive communication & adoption of best practices throughout the open source ecosystem. Bob is a member of the Technical Advisory Council for sigstore, a Linux Foundation / OpenSSF set of projects focused on improving transparency and UX of software supply chains. Before joining Google in 2021, Bob was a member of Red Hat's Office of the CTO where he was responsible for emerging technology strategy with strategic partners (including IBM) and a principal architect at NetApp where he focused on contributions to OpenStack and storage automation projects. He holds a PhD in Computer

Engineering from NC State University where he also serves as an adjunct assistant professor in the ECE department.

Mel Chua is a contagiously enthusiastic hacker, scholar, and perpetual motion machine. She is an auditory low-pass filter and multimodal polyglot and a PhD candidate at Purdue University's School of Engineering Education. Mel received her B.S. in Electrical and Computer Engineering from Olin College of Engineering and spent several years in the open-source software and hardware industry before returning to academia. Mel's research focuses on faculty development, learning in hacker/maker communities, embodied qualitative research methodologies, and prototyping alternate ontologies of curricular culture in engineering education.

Lin Clark is a Senior Principal Engineer and Acting Director of the WebAssembly team at Fastly. She is also the chair of the W3C's WebAssembly System Interface (WASI) subgroup. In her 15+ years in open source, she has been a maintainer on Firefox developer tools, worked as an early employee at npm, and has been a core module maintainer on Drupal, among other things. She also has a long running series of explainers called Code Cartoons which have explored many security topics, from DNS over HTTPS to the capability-based security model of WASI.

Dr. **Sol Greenspan** serves as the program director for Software Engineering research and also manages the Software Security portfolio for the Secure and Trustworthy Cyberspace program. He leads a relatively new program on Designing Accountable Software Systems and also leads the Trustworthy AI theme of the National AI Research Institutes program. Prior to NSF, Dr. Greenspan conducted R&D in industrial research labs (Bell Laboratories, GTE Laboratories, Schlumberger-Doll Research) and has also taught and performed research at several universities. Dr. Greenspan received his Ph.D. degree in Computer Science from the University of Toronto for work in the intersection of Artificial Intelligence and Software Engineering. His M.S. in Computer Science is from Rutgers University, where he did an early project on machine learning, previously earning a B.S. in Mathematics from the University of Michigan.

Luke Hinds works within the Emerging Technologies group in Red Hat's CTO office, where he leads a team working on open source security. Luke is the founder of project sigstore and has held numerous community roles, such as the Kubernetes Security Response Team, elected member of the Open Source Security Foundation Technical Advisory Council and is a board member of the confidential computing foundation

Justin Hutchings is the Director of Product Management for Supply Chain Security at GitHub where he works on products like Dependabot and the GitHub Advisory Database. He has extensive experience in open source and standards development and has contributed to initiatives in the Open Software Security Foundation (OpenSSF), Open Connectivity Foundation, IEEE, and USB-IF. Prior to joining GitHub, he was a product manager at Microsoft where he built developer platforms as part of Azure Identity, Microsoft Research, and Windows. Justin earned his BS in Software Engineering and Computer Science from Rose-Hulman Institute of Technology.

Dustin Ingram is a software engineer on Google's Open Source Security Team, where he works on improving the security of open-source software that Google & the rest of the world relies on. He's also a director of the Python Software Foundation, and maintainer of the Python Package Index.

Greg Kroah-Hartman is among a distinguished group of software developers who maintain Linux at the kernel level. In his role as a Linux Foundation Fellow, he continues his work as the maintainer for the Linux stable kernel branch and a variety of subsystems while working in a fully neutral environment. He also works closely with Linux Foundation members and projects, and on key initiatives to advance Linux.

Dr. **Uma Karmarkar** is an Assistant Professor with a dual appointment between the Rady School of Management and the School of Global Policy and Strategy at the University of California, San Diego. Prior to this, she was a member of the Marketing Unit faculty at the Harvard Business School and affiliated with the Harvard Center for Brain Science. She holds dual PhDs in neuroscience and consumer behavior. Dr. Karmarkar is a neuroeconomist whose research draws on neuroscience, psychology, behavioral economics and marketing to develop interdisciplinary frameworks of applied decision-making. Reflecting this background, her work has been published in academic journals ranging from Neuron to Management Science, and covered by popular media outlets including Scientific American, The Economist, and The New York Times.

Per Larsen leads a security consultancy (Immunant, Inc.) focusing on hardening systems software against memory corruption vulnerabilities. He is particularly interested in compile- and runtime techniques that drive up the cost of exploitation as well as efforts to migrate privileged, low-level code to safe and modern languages such as Rust. He is responsible for the C2Rust effort which aims to automate safety-enhancing source code translation.

Joshua Lock is a Staff 2 Open Source Engineer in VMware's Open Source Program Office where he works on software supply chain security standards and tools. He is a steering committee member and maintainer for the Supply chain Levels for Software Artifacts (SLSA) project, an editor of The Update Framework (TUF) specification and maintainer of python-tuf and go-tuf implementations, and a root key holder for and contributor to Sigstore. Joshua has a long history of contributing to open-source software. His noted works to date are on build tools (Yocto Project, OpenEmbedded), CI/CD systems, Linux distributions (MeeGo, Moblin, Tizen), UX for clamshell and tablet devices (GNOME), and more that he can't remember.

Bob Lord joined the Cybersecurity and Infrastructure Security Agency (CISA) as a Senior Technical Advisor in April 2022. Previously he was the Chief Security Officer at the Democratic National Committee where he brought more than 20 years of experience in the information security space to the committee, state parties, and campaigns. Before that he was Yahoo's Chief Information Security Officer, covering areas such as risk management, product security, security software development, e-crimes and APT programs. He was the Chief Information Security Officer in Residence at Rapid 7, and before that headed up Twitter's information security program as its first security hire.

Anil Madhavapeddy is Professor of Planetary Computing at the Department of Computer Science & Technology at the University of Cambridge. His research covers the intersection of large-scale systems and robust programming methods such as functional programming. He has worked on open-source systems since the 90s, and has been a maintainer on OpenBSD, Docker, Xen and OCaml (where he co-developed the opam package manager). He is a founding director of the Cambridge Centre for Carbon Credits, which aims to halt tropical deforestation by developing a robust carbon offset mechanism using global satellite data.

Nicholas Matsakis is a Senior Principal Engineer at AWS and co-lead of the Rust language design team. He has been working on Rust since 2011, with a focus on its type system and compiler implementation. He did his undergraduate study at MIT, graduating in 2001, and later obtained a Ph.D. in 2011, working with Thomas Gross at ETH Zurich.

Shane Miller is chair of the Rust Foundation, a founding member of the Rust Foundation board of directors, and the leader of Rust open source at Amazon Web Services (AWS). The Rust programming language combines the performance and resource efficiency of systems programming languages like C with memory safety, eliminating a substantial class of high severity security issues. During Shane's tenure as a Rust leader, the community has nearly quadrupled (from 600,000 to 2.2MM developers worldwide). Over the last three decades, Shane's held diverse roles, including principal engineer, university faculty, and political consultant. Shane's engineering experience includes insurance, globalization, machine learning, cryptography, programming languages, and open source. She holds B.S. and M.S. degrees in pure mathematics.

Daniela Oliveira is a Program Director at the NSF Computer and the Directorate of Information Science and Engineering (CISE), Division of Computer and Network Systems (CNS), Secure and Trustworthy Cyberspace (SaTC), where she focuses on the Systems portfolio. She received her B.Sc. and M.Sc. degrees in Computer Science from the Federal University of Minas Gerais in Brazil. She then earned her Ph.D. in Computer Science from the University of California at Davis. She is on rotation from the University of Florida, where she is an Associate Professor at the Department of Electrical and Computer Engineering, where she specializes on socio-technical aspects of cyber security systems research, including malware analysis and detection, cyber social engineering (phishing and mis/disinformation), and developer blindspots while coding. Daniela Oliveira received a National Science Foundation CAREER Award in 2012 for her innovative research into operating systems' defense against attacks using virtual machines, the 2014 Presidential Early Career Award for Scientists and Engineers (PECASE) from President Obama, and the 2017 Google Security, Privacy and Anti-Abuse Award. She is a National Academy of Sciences Kavli Fellow and a National Academy of Engineers Frontiers of Engineering Symposium Alumni. Her research has been sponsored by the National Science Foundation (NSF), the Defense Advanced Research Projects Agency (DARPA), the National Institutes of Health (NIH), the MIT Lincoln Laboratory, and Google. While serving the NSF she received the 2022 Director's Award for Superior Accomplishment (Group) for contributions to the Resilient and Intelligent NextG Systems (RINGS) program.

Deborah Shands is a security researcher and senior computer scientist at SRI International, where she currently focuses on security and privacy for digital credential wallets and decentralized identities. Prior to joining SRI, she assessed security architectures and designs for space systems at The Aerospace Corporation and served as a Program Director for the National Science Foundation. Her research has focused on the security of distributed computing environments, including scalable security administration for distributed systems, mission-oriented access control in coalition environments, as well as trust establishment for system component integration.

Deian Stefan is an Associate Professor of Computer Science and Engineering at UC San Diego, where he co-leads the Security and Programming Systems groups. His research lies at the intersection of security and programming languages, with a particular focus on building secure systems that can be deployed in production. Deian is on the Bytecode Alliance board of directors, serves on several industry security working groups, and co-founded two companies (Intrinsic (acquired by VMWare) and Cubist). His work has been recognized by multiple awards, including distinguished paper awards, the NSF CAREER award, and the Sloan Fellowship.

Camille Stewart is the inaugural Deputy National Cyber Director for Technology and Ecosystem Security in the Executive Office of the President for the Biden-Harris Administration. Prior to taking this role Camille was a security leader at Google. She was the Global Head of Product Security Strategy at Google advising Google's product leads on federated security and risk. She also led security, privacy, election integrity, and dis/mis-information for Google's mobile business as the Head of Security Policy for Google Play and Android. Prior to Google, Camille was a manager in Deloitte's Cyber Risk practice working on cybersecurity, election security, tech innovation, and risk issues for DHS, DOD, and other federal agencies. Camille is the former Senior Policy Advisor for Cyber, Infrastructure & Resilience Policy at the Department of Homeland Security. Appointed by President Obama, Camille contributed to a number of federal cyber policies such as Presidential Policy Directive 41 (PPD -41) on United States Cyber Incident Coordination and Cybersecurity National Action Plan (CNAP). Camille focused on a number of domestic and international cyber and technology policy issues, earning recognition from President Obama for her contributions to expanding cybersecurity cooperation with DHS's Israeli counterparts. Prior to working at DHS, Camille spent five years as the Senior Manager, Legal Affairs at Cyveillance, Inc., a cybersecurity company focused on open source threat intelligence and incident response (now ZeroFOX). While there, Camille navigated legal and policy challenges for cyber-related issues such as data privacy, incident response, Internet governance,

cyber security, new gTLDs, social media law & policy, and intellectual property (IP) protections online for Global 2000 companies. In this role, Camille managed a team of cyber intelligence analysts in the SOC, revamped the brand protection service offerings, managed the company's policy and IP portfolio, and built new incident response service offerings. Camille also spent time working for Rep. Marcia Fudge and Rep. Emanuel Cleaver II. Camille is passionate about making technology and IP issues accessible to entrepreneurs and the less than technically savvy. To that end, she founded a legal consultancy and startup incubator, MarqueLaw, PLLC, and the blog **TheDigitalCounselor.com**.

David Tarditi is Vice President of Engineering at CertiK, a blockchain security company. At CertiK, he leads work on tools for securing blockchain programs and smart contracts. Prior to joining CertiK, he had a 25-year career at Microsoft. His last role was leading the development of a secure IoT operating system. Other roles included researcher, development manager, development lead, and research group manager. David has worked on compilers, programming languages, operating systems, and security. He is an expert at building software development tools for enabling the development of secure software. He created the Checked C extension for C, which enables more secure C code to be written. He led the creation of an ahead-of-time compiler for C# so that C# could be used for systems software development. David has a bachelor's degree in engineering from Princeton University and a Ph.D. in Computer Science from Carnegie Mellon University.

Marshall Van Alstyne (@InfoEcon) is coauthor of the international bestseller **Platform Revolution**. He is one of the world's foremost experts on network business models and is the Questrom Chair Professor of Management at Boston University. He is a frequent speaker, board advisor, and consultant to startups and global firms. In 2019, Thinkers 50 named him one of the top management thinkers globally. His research has received more than 20,000 citations, a dozen academic awards, and appeared in such places as *Science*, *Nature* and *Strategic Management Journal*. Interviews appear regularly across *Bloomberg*, The *Economist*, The *New York Times*, The *Wall Street Journal* and *National Public Radio*. He studied computer science at Yale and information economics at MIT. He holds multiple patents; was among the first to measure the dollar value of social networks. Marshall is a husband and dad, who loves dogs, exercise, travel, and questions of governance.

Dr. **Sam Weber** has worked in government, academia, and industry. Currently he is a Program Officer at the Office of Naval Research and has been a Program Director at the National Science Foundation. Previously he has been a faculty member at Cornell University and the University of Pennsylvania, and a Research Staff Member at IBM's TJ Watson Research Center.