12-19-2023

Shields Up For Software

Derek E. Bambauer
University of Florida Levin College of Law, bambauer@law.ufl.edu

Melanie J. Teplinsky
American University Washington College of Law, teplinsk@american.edu

SHIELDS UP FOR SOFTWARE

LAWFARE (Dec. 19, 2023)
https://www.lawfaremedia.org/article/shields-up-for-software

Derek E. Bambauer & Melanie J. Teplinsky[*]

Table of Contents

[*] Irving Cypen Professor of Law, University of Florida Levin College of Law, bambauer@law.ufl.edu, and Senior Fellow, Tech, Law, and Security Program, American University Washington College of Law, teplinsk@american.edu. Authors are listed alphabetically. We thank Saumya Debray, Jim Dempsey, James Halpert, Paul Rozensweig, Paul Tiao; the Lawfare editors; and the participants in a series of working group sessions on software security for helpful feedback.

INTRODUCTION

As part of the National Cybersecurity Strategy,[1] the Biden administration seeks to "develop legislation establishing liability for software products and services,"[2] which would include "an adaptable safe harbor framework to shield from liability companies that securely develop and maintain their software products and services."[3] We propose that this software liability regime incorporate one safe harbor and one "inverse safe harbor." The first would shield software creators and vendors from liability if they follow enumerated best practices in design, development, and implementation. The second—the inverse safe harbor, or sword—would automatically impose liability on developers who engage in defined worst practices.[4] The safe and inverse safe harbors will provide certainty to regulated entities, reduce administrative costs, and create incentives for improving security. This article describes the twin safe harbors, their policy goals, and the key design criteria for their success.

I. POLICY RATIONALES

At present, software is almost entirely exempt from the usual liability regimes that apply to products and services. Tort law excludes software for fear that imposing liability would cripple innovation.[5] In addition, tort's economic loss doctrine prevents recovery for pecuniary harm not linked to physical damage or injury, creating a major obstacle to recovery for harms caused by flawed software.[6] Contract law enables software vendors to disclaim warranties and bar users from litigation even if they suffer harm caused by code.

The administration's cybersecurity strategy—as applied to software liability—would alter this status quo. A well-constructed liability regime would improve cybersecurity and penalize shoddy code while protecting software producers who follow reasonable practices. However, the new system will create risk, especially from uncertainty, for those who create and distribute code.

---

[1] NATIONAL CYBERSECURITY STRATEGY (March 2023).
[2] *Id.* at 21.
[3] *Id.*
[4] *See* Derek E. Bambauer, *Cybersecurity for Idiots*, 106 MINN. L. REV. HEADNOTES 172 (2021).
[5] *See* Michael D. Scott, *Tort Liability for Vendors of Insecure Software: Has the Time Finally Come?*, 62 MD. L. REV. 425 (2008).
[6] *See* Danielle K. Citron, *Reservoirs of Danger: The Evolution of Public and Private Law at the Dawn of the Information Age*, 80 S. CAL. L. REV. 241 (2007).

The proposed software liability regime rests on a standard of reasonable care for design, development, and maintenance of code. Failure to meet the standard of care is negligence. This standard-of-care approach is widely used and often framed as a reasonableness inquiry. For example, in medical malpractice cases, the criterion is typically how a reasonable physician would have treated the patient. Standards of care are helpfully flexible and usually consider each case's individual circumstances. However, this fact-specific analysis requires gathering considerable information and is thus costly. Decision-makers also inevitably make mistakes. Some actors who fail to meet the standard of care will escape liability, and some who conform to it will be punished. Safe harbors can improve this design. They provide certainty to regulated entities: immunity for following defined best practices, and clear liability for engaging in terrible ones. And safe harbors shape incentives: Coders and vendors that cannot or will not improve their code and business practices will be pushed out of the market.

II. SAFE HARBORS: AUTOMATIC IMMUNITY, AUTOMATIC LIABILITY

We propose augmenting the standard of care for secure software with two safe harbor provisions: one that automatically imposes liability for defined worst practices, and one that automatically confers immunity for following enumerated best practices.

A. The Sword: Automatic Liability

First, the liability regime should include minimum requirements for software development by defining a set of worst practices that automatically create liability (the "inverse safe harbor," or sword). For example, including a hard-coded account with administrative privileges and a password that cannot be changed is clearly an unacceptable decision.[7] When hard-coded credentials[8] are used in operational technology products[9] in critical industries such as power, for example, an adversary who discovers the password could change how the installed products operate. Developers who are careless or indifferent to security should face significant penalties. Risks of

---

[7] *See, e.g.*, *D-Link Agrees to Make Security Enhancements to Settle FTC Litigation*, FED. TRADE COMM'N (July 2, 2019) (announcing settlement with smart home products manufacturer that distributed software "using hard-coded login credentials… with the easily guessed username and password, 'guest'").

[8] *See CWE-798: Use of Hard-coded Credentials*, COMMON WEAKNESS ENUMERATION (last updated Oct. 26, 2023).

[9] *See* Christian Vasquez, CISA urges vendors to get rid of default passwords, CYBERSCOOP (Dec. 15, 2023).

overdeterrence are minimal: There is no reasonable basis for engaging in worst practices.

The inverse safe harbor should begin by including practices for which there is widespread consensus that they are dangerous. We suggest starting with the bad coding practices from the Common Weakness Enumeration[10] list managed by the Cybersecurity and Infrastructure Security Agency (CISA).[11] These are programming or design choices that are indefensible, albeit at differing levels of severity. Over time, the list of worst practices should grow. Indeed, decisions about compliance with the standard of care will likely identify conduct sufficiently far below the standard that it merits inclusion as a worst practice. Security-related litigation and settlements by the Federal Trade Commission (FTC)[12] pursuant to its Section 5 authority[13] could also provide examples[14] of sufficiently bad practices[15]—since the FTC must often make determinations about[16] the adequacy of organizations' security practices[17].

B. The Shield: Automatic Immunity

Second, the affirmative safe harbor defines conduct that makes developers immune from liability for particular features and design choices. The immunity safe harbor includes best practices: coding and design choices that maximize security and minimize risk. Immunity should be issue specific; software might comply with some best practices but not others, and the liability shield would apply only to compliant aspects. Developers will seek to incorporate these best practices because doing so guarantees freedom from liability—if not for other, more laudable reasons. Safe harbor provisions are common in regulatory regimes. For example, the safe harbors for online service providers[18] created by the Digital Millennium Copyright Act (DMCA) ensure these entities cannot be held liable for transporting or storing content that infringes copyright if they follow prescribed rules, such as

---

[10] *See* Common Weakness Enumeration.
[11] We thank Professor Saumya Debray of the University of Arizona Department of Computer Science for discussion of this example.
[12] *See* Daniel J. Solove & Woodrow Hartzog, *The FTC and the New Common Law of Privacy*, 114 COLUM. L. REV. 583 (2014).
[13] *See* Justin (Gus) Hurwitz, *Data Security and the FTC's UnCommon Law*, 101 IOWA L. REV. 955 (2016).
[14] *See* Cory Bennett, *Software firm settles FTC charges it misled on encryption*, THE HILL (Jan. 6, 2016).
[15] *See* Cory Bennett, *Oracle settles FTC charges that it deceived with security updates*, THE HILL (Dec. 22, 2015).
[16] *See* LabMD v.FTC, 894 F.3d 1221 (11th Cir. 2018).
[17] *See* FTC v. Wyndham Worldwide Corp., 799 F.3d 236 (3d. Cir. 2015).
[18] Codified at 17 U.S.C. § 512.

removing access to infringing material on notification. Nearly all service providers comply[19] with the DMCA safe harbor requirements because they value the legal certainty immunity provides.

The immunity safe harbor should contain two types of requirements. The first are process-based mandates,[20] which require organizations to follow defined steps or protocols as part of developing and maintaining software code. For example, organizations might be required to implement a vulnerability-reporting mechanism and to document how they respond to "bug reports"[21] they receive. The second are substantive provisions,[22] which are specific requirements for features, design decisions, and the like. For example, developers might have to create programs that automatically encrypt sensitive data using a cipher or system with defined minimum strength[23] (for example, quantum-resistant encryption[24]). American cybersecurity regulation tends to concentrate on process to the exclusion of substance, to the detriment of software security and cybersecurity more generally.[25] Process-based approaches can help ensure consistency across an organization and help avoid inadvertent errors that compromise security. However, they are less effective in ensuring that the process's outcome is sufficiently secure. Some security issues are amenable to substantive rules.[26] Data should be encrypted. Memory-safe languages should be used. Data inputs should be checked for their type and length[27] to prevent buffer overflow attacks[28]. Other issues are highly contextual and better managed by standards or by process obligations.

Enumerating best practices for the safe harbor will take time and may be context dependent. William McGeveran, professor of information law and data privacy at the University of Minnesota Law School, offers an analogous approach in the cybersecurity context, suggesting that an initial set of best practices could be drawn from the requirements included in most or all extant

---

[19] *See* Jennifer M. Urban, Joe Karaganis, & Brianna Schofield, Notice and Takedown in Everyday Practice, UC BERK. PUB. L. RES. PAPER NO. 2755628 (Mar. 24, 2017).
[20] *See Information security, cybersecurity and privacy protection*, ISO/IEC 27001:2022.
[21] *See* Shreya Bose, *How to write an Effective Bug Report*, BROWSERSTACK (Dec. 21, 2022).
[22] *See, e.g.,* 45 C.F.R. § 164.312.
[23] *See FIPS PUB 140-3: Security Requirements for Cryptographic Modules*, Nat'l Inst. Standards & Tech., Dep't of Commerce (Mar. 22, 2019).
[24] *See NIST Announces First Four Quantum-Resistant Cryptographic Algorithms*, Nat'l Inst. Standards & Tech., Dep't of Commerce (July 5, 2022).
[25] *See* Derek E. Bambauer, *Ghost in the Network*, 162 U. PA. L. REV. 1011 (2014).
[26] *See* Derek E. Bambauer, *Rules, Standards, and Geeks*, 5 BROOK. J. CORP. FIN. & COMM'L L. 49 (2010).
[27] *See Buffer Overflow*, OWASP.
[28] *See CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')*, COMMON WEAKNESS ENUMERATION (last updated Oct. 26, 2023).

certification frameworks for cybersecurity.[29] Software security best practices could be mapped in similar fashion. Guidance from industry leaders, such as Microsoft with its Security Development Lifecycle[30] and Google with its Cloud security best practices,[31] can offer candidates for inclusion. Finally, we believe that the immunity safe harbor should begin with requirements that apply to all software code; though as the safe harbor develops, it may be appropriate for certain mandates to be sector specific.

III. UPDATES

The lists for both safe harbors will require frequent revision. We suggest that the CISA be tasked with ongoing, regular information-gathering and rulemaking to update the twin safe harbors. The agency should draw on industry-specific knowledge from expert sources, ranging from federal agencies to private trade organizations. The Business Software Alliance[32] can offer input about general software development best practices, while the sector risk management agencies and information sharing and analysis centers[33] will have industry-specific insights. With respect to agencies, the Department of Health and Human Services[34] (in its role enforcing the Security Rule[35] established by the Health Insurance Portability and Accountability Act of 1996) and the Federal Aviation Administration[36] (in its role ensuring safe air travel[37]) will have invaluable domain-level expertise and networks.

Ideally, the CISA would be authorized to conduct both public and private consultations, followed by an abbreviated notice-and-comment period, to issue annual updates. This design employs a co-regulatory model,[38] in which industry expertise undergirds and strengthens regulatory oversight. In addition, the CISA should have an emergency capacity to add to the

---

[29] *See* William McGeveran, *The Duty of Data Security*, 103 MINN. L. REV. 1135, 1141-93 (2019).
[30] *See* MICROSOFT SECURITY DEVELOPMENT LIFECYCLE (SDL).
[31] *See Google Cloud security best practices center*, GOOGLE.
[32] *See https://www.bsa.org/reports/updated-bsa-framework-for-secure-software*, BSA.
[33] *See Nat'l Council of ISACS* (last visited Feb. 17, 2024).
[34] *See Security Rule Guidance Material*, Dep't of Health & Human Servs. (last updated Feb. 16, 2024).
[35] *See The Security Rule*, Dep't of Health & Human Servs. (last updated Oct. 20, 2022).
[36] *See* LOOKING AHEAD AT THE CYBERSECURITY WORKFORCE AT THE FEDERAL AVIATION ADMINISTRATION, NAT'L ACADS. OF SCI., ENG'G, AND MED. (2021).
[37] *See* AVIATION CYBERSECURITY: FAA SHOULD FULLY IMPLEMENT KEY PRACTICES TO STRENGTHEN ITS OVERSIGHT OF AVIONICS RISKS, GOV'T ACCOUNTABILITY OFF. (Oct. 9, 2020).
[38] *See* David Thaw, The Efficacy of Cybersecurity Regulation, 30 GA. ST. U. L. REV. 287 (2014).

inverse safe harbor when new information about highly risky practices emerges. Last, the CISA should be empowered with a confidential reporting function[39] so that software developers, testers, and other experts could report risks and vulnerabilities without fear[40] this material could be disclosed in litigation or used for regulatory enforcement action. The Cybersecurity Information Sharing Act of 2015 could serve as a useful model in this regard.[41] Its provisions prohibit cybersecurity threat information shared with the government pursuant to the act from being used by the government as the basis for a regulatory or enforcement action against the entity that shared the information. Reporting could be made mandatory in some circumstances, as it already is for data breaches in many instances.[42]

IV. RISKS

The safe harbors' certainty could have undesirable consequences. For example, software developers may have little incentive to take security precautions greater than the immunity safe harbor's requirements, which could effectively convert those mandates into a ceiling. The liability regime can mitigate this risk by establishing safe harbor requirements that are sufficiently rigorous to ensure strong security even if developers treat them as sufficient rather than necessary.

V. KEY DESIGN CHOICES

Here we highlight six key system design choices for the safe harbors.

A. Deterrence

Selecting requirements for the immunity safe harbor (the shield) is critical for at least two related reasons. First, if a developer follows a given requirement, the threat of legal liability for that aspect of the code disappears. Choosing the correct level of precautions is vital to setting incentives correctly: If the standard is too demanding, providers may overinvest in it to the detriment of other security measures; if it is too lax, providers can immunize themselves despite supplying code with subpar security. Second, for many software providers, the safe harbor will function as the de facto

---

[39] *See* Derek E. Bambauer, *Ghost in the Network*, 162 U. PA. L. REV. 1011, 1086-87 (2014) (describing similar confidential reporting systems).
[40] *See* Derek E. Bambauer & Oliver Day, *The Hacker's Aegis*, 60 EMORY L.J. 1051 (2010).
[41] *See* Brad S. Karp, *Federal Guidance on the Cybersecurity Information Sharing Act of 2015*, HARV. L. SCHOOL FORUM ON CORP. GOVERNANCE (Mar. 3, 2016).
[42] *See Data Breaches*, PRIVACYRIGHTS.ORG (Jan. 27, 2023).

practical standard of care, because these entities will not want to expose themselves to legal risk through noncompliance. Practically speaking, the demands of the immunity safe harbor will be the rules of the road for major software developers.

B. Legacy Code

Both safe harbors must address legacy code. Software standards and coding practices change over time, hopefully for the better. We suggest a bifurcated approach to legacy code. The immunity safe harbor should apply immediately on adoption to any code, of any age—compliance creates immunity.

By contrast, the penalties of the inverse safe harbor (sword) should be implemented with a time lag. Developers did not have notice of these requirements and their legal effect when designing older code, and they will need time to bring it up to compliance. Computer science studies[43] suggest the average software life cycle is six to eight years[44]. We suggest a time lag for implementation of the inverse safe harbor half as long: three years. Microsoft, for example, supports a given Service Pack of major fixes to its Windows operating system versions for 12-24 months, with a standard support cycle of five years.[45] The CISA, or industry-specific security regulators, should be empowered to grant waivers for up to two additional years on showing of significant need or hardship. For example, the software that runs most ATMs in the United States is written in COBOL because that language runs efficiently on mainframe computers and is optimized for business functions. However, there are very few remaining COBOL programmers, making replacing that code difficult; in fact, IBM has begun to use artificial intelligence capabilities to convert COBOL to Java.[46]

C. Critical Infrastructure

Software in critical infrastructure might be held to a more demanding set of best practices to obtain immunity. Specialized treatment for critical infrastructure systems could drive progress toward greater security in a context where harm from insecurity would be particularly severe. One challenge is that the definition of "critical infrastructure" has expanded

---

[43] *See* M.M. Lehman, *Programs, life cycles, and laws of software evolution*, 68 PROC. OF IEEE 1060 (1980).
[44] *See* Sam Williams, *A unified theory of software evolution*, SALON (Apr. 8, 2002).
[45] *See Fixed Lifecycle Policy*, MICROSOFT (Feb. 21, 2023).
[46] *See* JD Sartrain, *The World Depends on 60-Year-Old Code No One Knows Anymore*, PC MAG. (Dec. 1, 2023).

significantly over time:[47] A 2003 presidential directive on the subject included national monuments and icons within its ambit, where they remain today[48]. When critical infrastructure is everything, it is nothing.[49] It is possible, if not likely, these more rigorous standards will generate industry resistance. The more focused definition of a "covered entity" (that is, ones of higher criticality) under the Cyber Incident Reporting for Critical Infrastructure Act of 2022 (CIRCIA),[50] as currently contemplated by the CISA pending its final rulemaking,[51] could be an appropriate starting point for specialized best practices if they are deemed necessary. Another possible starting point would be the "National Critical Functions" as defined by the CISA.[52] These are the functions deemed so vital to the country that their disruption would have a "debilitating effect" on national security, national economic security, national public health or safety, or any combination thereof. However, regulators should evaluate the progress made under the new software liability regime generally before deciding whether a bespoke subsystem for critical infrastructure is necessary. If it is, the tailored immunity safe harbor regime should apply only to the most vital infrastructure contexts, such as those defined under the CIRCIA.

D. Testing

Determining compliance with the immunity safe harbor is challenging, but critical. There are essentially three options: evaluation after the fact (such as post-breach), self-certification, or external testing.

Liability based on ex-post analysis is a core feature of the standard-of-care approach, so we do not discuss it further here. Self-certification risks inadequate investigation and concealment, although there are examples of mechanisms that could reduce these inherent problems (such as the required certification of a firm's internal controls[53] under Section 404 of the Sarbanes-Oxley Act,[54] backed by potential civil and criminal penalties, or the

---

[47] *See* Derek E. Bambauer, *Conundrum*, 96 MINN. L. REV. 584, 643 (2011).
[48] *See* John Moteff & Paul Parfomak, *Critical Infrastructure and Key Assets: Definition and Identification*, CRS REPORT FOR CONG. (Oct. 1, 2004).
[49] *See* Will Loomis, *Modernizing critical infrastructure protection policy: Seven perspectives on rewriting PPD21*, ATL. COUNCIL (Mar. 22, 2023).
[50] Pub. L. No. 117-103, Div. Y, 135 STAT. 49, 1038 (117th Cong. 2022) (to be codified at 6 U.S.C. § 681 et seq.).
[51] *See Request for Information on the Cyber Incident Reporting for Critical Infrastructure Act of 2022*, 87 FED. REG. 55833 (Sept. 12, 2022).
[52] *See National Critical Functions Set*, CISA (Apr. 2019).
[53] *See Study of the Sarbanes-Oxley Act of 2002 Section 404 Internal Control over Financial Reporting Requirements*, U.S. SECURITIES & EXCH. COMM'N (Sept. 2009).
[54] *See* Katie Terrell Hanna, *Sarbanes-Oxley Act (SOX) Section 404*, TECHTARGET (Mar. 2022).

Department of Justice's Civil Cyber-Fraud Initiative[55]). External testing is undoubtedly the most effective option, although also likely the most expensive. However, establishing effective external verification, especially regarding the independence of auditors, is a significant challenge.

We suggest that developers initially be allowed to choose a mechanism for compliance determinations. This could generate a market: from less rigorous and costly self-certification to more searching but authoritative inspection by an expert private organization, such as cybersecurity firms or the consulting arms of accounting firms, or a government entity, such as the CISA or an industry-specific regulator. This choice itself could create incentives, since the selection of compliance certification mechanism could provide an honest signal to software consumers and to regulators about the security of a particular piece of code. Furthermore, auditing of publicly traded firms, supervised by the Securities and Exchange Commission, provides a well-working model for this testing requirement.

If accumulated experience shows that this menu of certification mechanisms does not provide sufficiently reliable information, government auditors could step in as the permanent testing body. The CISA might establish an auditing function, much as the Federal Reserve does for the banking system.[56] In any case, we believe that testing and auditing of code that seeks immunity is critical to the safe harbor and to the liability regime itself.

E. Transitional and Other Support

Compliance with the proposed software liability regime will impose costs on developers, who will not be able to fully capture the benefits of the resulting improved security. Software security is thus a classic example of a positive externality.[57] This raises two concerns: transition costs, and effects on small and medium-sized businesses (SMBs), including many open-source software (OSS) organizations and academic institutions. We propose two complementary interventions: providing carrots and reducing the size of the stick.

Greater software security achieved through the proposed legal liability system will benefit society generally. The software entities that

---

[55] *See* Cynthia Brumfield, *Cyber-related False Claims actions are on the uptick*, CSO (Sept. 18, 2023).

[56] *See Understanding Federal Reserve Supervision*, BD. OF GOVERNORS OF FED. RESERVE BANK. SYS. (Apr. 27, 2023).

[57] *See Externalities - The Economic Lowdown Podcast Series*, FED. RESERVE BANK OF ST. LOUIS.

provide this benefit, potentially at significant cost, will not earn enough revenues to offset those expenditures. Government intervention is one classic answer to externality problems. If we want software providers to undertake costly action that provides broadly shared benefits, government should subsidize their efforts. We suggest either direct subsidies to software developers or tax credits for expenses linked to compliance. However, these costs will diminish with time and may well diminish rapidly as developers learn to implement best practices (and avoid worst ones) more efficiently. We propose that transitional financial support from government last for the same period of time as the exemption for legacy code: three years, with up to two additional years for cause on an individual basis. The federal government might also speed replacement of less secure legacy code by expanding financial support if a software entity moves more quickly to remediate its existing products. For example, a firm that replaces older, substandard code in two years rather than the allotted three might be granted funding equal to the time saved—here, an additional year of support (four years rather than three).

Similarly, if concerns that the immunity safe harbor creates a ceiling for security materialize in practice, funding could create incentives for developers to follow preferred best practices that implement additional safeguards.[58] For example, if the immunity safe harbor did not mandate that developers use memory-safe languages, additional government funding could be conditioned on utilizing them.[59] Such support also creates incentives for software developers to implement precautions efficiently based on private information and expertise.[60]

Reducing the size of the stick could also help SMBs, including OSS organizations that are essentially volunteer collectives, by ensuring that they do not cease development due to liability risks. One mechanism for limiting disincentives would be to set a liability cap: a maximum amount of damages for which an SMB could be held responsible, either as a flat ceiling (based on the size or revenues of the entity) or as a multiple of profits, revenues, or the like. This creates a risk of under-deterrence, since organizations may not face liability equal to the potential or expected harm they generate. Thus, we believe if a liability cap is implemented, it should not apply to software used exclusively or primarily in critical infrastructure. (We include the limiting language "exclusively or primarily" because otherwise most significant software programs, including staples such as operating systems and web

---

[58] *See* Franklin D. Kramer, Melanie J. Teplinsky, & Robert J. Butler, *Cybersecurity for innovative small and medium enterprises and academia*, ATL. COUNCIL (Jan. 25, 2022).
[59] *See Software Memory Safety*, NAT'L SEC'Y AGENCY (Nov. 2022).
[60] *See* Franklin D. Kramer, Melanie J. Teplinsky, & Robert J. Butler, *We need a cybersecurity paradigm change*, THE HILL (Feb. 15, 2022).

browsers, would be drawn within this exception, thereby vitiating it.) However, developers need not face the full social cost of insecurity to have adequate incentives—a level of potential damages that forces market exit or bankruptcy is likely to suffice. Liability caps may also be necessary to ensure the survival of many software SMBs, including startups. Small businesses, open-source organizations, and academic institutions tend to be wellsprings of innovation[61] and receive support based on that generativity under other government programs, such as patent policy[62].

F. Mechanism

The optimal mechanism to implement the safe harbors, and the software liability regime more generally, is comprehensive federal legislation. However, aspects of our proposal could be partially implemented through alternative means. For example, compliance with the safe harbor requirements could be a condition of eligibility for certain software and information technology purchases by the federal government, such as those imposed via the Federal Acquisition Regulation (FAR)[63] and Defense Federal Acquisition Regulation Supplement (DFARS)[64]. Or compliance could be encouraged, yet formally made optional, by use of federal subsidies or tax credits without implementation of the liability regime. State and local governments could undertake similar actions. While piecemeal measures are second best, they would nonetheless improve on the current situation.

---

[61] *See* Franklin D. Kramer, Melanie J. Teplinsky, & Robert J. Butler, *Cybersecurity for Innovative Small and Medium Enterprises and Academia*, ATL. COUNCIL (Jan. 2022).
[62] *See Reducing Patent Fees for Small Entities and Micro Entities Under the Unleashing American Innovators Act of 2022*, 88 FED. REG. 17147 (Mar. 22, 2023).
[63] *See Federal Acquisition Regulation: Standardizing Cybersecurity Requirements for Unclassified Federal Information Systems*, 88 FED. REG. 68402 (Oct. 3, 2023).
[64] *See* 48 C.F.R. § 252.204-7012.

CONCLUSION

Including both an immunity safe harbor (the shield) and an inverse safe harbor (the sword) in the proposed software liability regime will make the system more predictable, effective, and efficient. These zones of immunity and absolute liability complement the standard of care and draw on distributed information about software design and implementation. Since under our proposal software entities that comply with the safe harbors would obtain significant reductions in costs, including expected liability risk, the requirements imposed would likely shape conduct in a direction that advances the goals of the National Cybersecurity Strategy.

* * *