Atlantic Council

CYBER STATECRAFT INITIATIVE

DESIGN QUESTIONS IN THE SOFTWARE LIABILITY DEBATE

by Maia Hamin, Sara Ann Brackett, and Trey Herr
with Andy Kotz

Atlantic Council

CYBER STATECRAFT INITIATIVE

The **Cyber Statecraft Initiative** works at the nexus of geopolitics and cybersecurity to craft strategies to help shape the conduct of statecraft and better inform and secure users of technology. This work extends through the competition of state and non-state actors, the security of the internet and computing systems, the safety of operational technology and physical systems, and the communities of cyberspace. The Initiative convenes a diverse network of passionate and knowledgeable contributors, bridging the gap among technical, policy, and user communities.

Authors

Maia Hamin
Sara Ann Brackett
Trey Herr

Editor

Samia Yakub

This report is written and published in accordance with the Atlantic Council Policy on Intellectual Independence. The author is solely responsible for its analysis and recommendations. The Atlantic Council and its donors do not determine, nor do they necessarily endorse or advocate for, any of this report's conclusions.

Atlantic Council
1030 15th Street NW, 12th Floor
Washington, DC 20005

For more information, please visit
www.AtlanticCouncil.org.

January 2024

Design by: Donald Partyka and Anais Gonzalez

Cover: A statue of Lady Justice is shown in front of several lines of code. Graphic designed by Anais Gonzalez and Donald Partyka. Photo credit goes to Tingey Injury Law Firm/Unsplash and Markus Spiske/Unsplash.

Atlantic Council

CYBER STATECRAFT
INITIATIVE

DESIGN QUESTIONS IN THE SOFTWARE LIABILITY DEBATE

by Maia Hamin, Sara Ann Brackett, and Trey Herr
with Andy Kotz

Acknowledgements

The authors would like to thank John Speed Meyers, Josephine Wolff, Bryan Choi and Melanie Teplinsky for their feedback on various versions of this document. We also thank the many unnamed individuals from government, academia, and industry who met with us and participated in events and group conversations on the topic of software liability, helping sharpen and clarify our thinking on the topics herein.

Table Of Contents

Executive Summary

Legal liability for insecure software is a deceptively simple-sounding concept that is, in practice, associated with a multifaceted and decades-long legal and policy debate. This paper identifies a set of core design questions for policy regimes to impose consequences on vendors of insecure software and surveys 123 articles from the wide-ranging literature on software liability to examine their viewpoints with respect to these key decisions.

The first design questions focus on what can create liability, often a combination of a failure to meet standards for good behavior with respect to the development and deployment of secure software and the manifestation of insecurity in software flaws that cause harm to a software user. These questions also raise the issue of responsibility—how to link the behavior of a software vendor to bad cybersecurity outcomes and account for the behavior of the software user with respect to software-specific practices such as patching. The next set of design questions focuses on the scope of a liability regime: whether it applies to all or only a subset of software, such as software that is used in high-risk sectors, that performs particular high-risk functions, that is produced by entities of a certain size, or software that is available for sale (versus released under an open source license). The third set of questions pertains to matters of governance and enforcement—how and by whom standards are defined, compliance assessed, violations prosecuted, and consequences determined.

In the sampled literature, certain legal questions—such as whether to favor tort liability based on product or negligence theories—have been much debated throughout the history of the surveyed literature with little evidence of an emerging consensus. Other questions, such as whether to hold software used in different sectors to different standards, or which specific security frameworks or practices to require, were less discussed. Some of these less-discussed questions, such as the question of whether to include developers of open source software in a liability regime, show relative consensus where they arise. Others, such as how to handle software patching, are disputed even in the more limited discussion that has occurred.

Perhaps the most important design question in the framework is that of the policy goal of such a regime. What problems within the existing software ecosystem does liability seek to correct? These potential goals, such as driving better ecosystem-wide security behavior or providing redress to harmed parties, often point in different directions with respect to how to resolve particular design questions in the construction of the regime. Debates, including those that have swirled in scholarly circles since the mention of software liability in last year's National Cybersecurity Strategy, have rarely articulated the full set of design questions available in the construction of a regime or explicitly mapped these questions to the goals of such an endeavor. This report concludes with a section using original analysis and the literature sample to examine how different design questions might be informed by the goals of a software liability regime.

Introduction

Legal liability has long been a solution proposed to fix markets in which buyers are ill-positioned to protect themselves through purchasing decisions or to rectify threats from too-dangerous products. Many today argue that the market for software security is broken in just such a way: makers of software face too little pressure from consumers to secure their software because consumers are ill-equipped to evaluate the security of such software itself and manufacturers pay few costs if their software is later found to be insecure. Legal scholars and cybersecurity researchers alike have long been interested in the idea of liability for insecure software, in hopes of providing redress for victims of insecurity or shifting incentives toward a better-secured software ecosystem. Following its mention in the National Cybersecurity Strategy,[1] the question of how to implement liability for vendors of insecure software is once again in the conversation.

However, the term liability itself and the goals that motivate it point not to a single type of legal regime but instead to a set of heterogenous policy constructs. Two broad buckets of such constructs are potential regimes based on torts versus potential regimes based on regulation. Torts allow one entity to sue another for "act[s] or omission[s] that cause legally cognizable harm to persons or property,"[2] and have evolved mostly through state standards and common law, or judges' rulings rather than explicit laws passed by federal lawmakers. However, Congress can pass laws that impact the implementation of torts, and many roads to software liability might involve a law that changes the way in which existing theories of tort have applied to software. In contrast, in a regulatory regime, a government body such as an expert agency defines standards and requirements for specific entities such as software vendors and then (often, though not always) enforces these requirements itself. Within both the broad buckets of torts liability and regulatory liability, there are different potential forms, from product- versus negligence-based torts to premarket approval requirements versus requirements to self-certify certain key practices with penalties for misrepresentation.

Thus, many questions remain about the form and nature of liability that would best achieve the goals laid out in the National Cybersecurity Strategy, and about the relative advantages of different potential paths to get there. These questions are not new, even if they are newly relevant; the debate over software liability has been evolving throughout academic research and writing, judicial opinion, and policy for almost as long as software has existed.

This report makes two contributions.

First, it deconstructs the liability debate into a set of policy design questions, and then, for each, identifies design options and models from existing legal structures that could be used to build and implement such element as well as articulating how each element relates to other questions and to the overall goals of such a regime. This framework deliberately uses terms that are different from the legal terms of art for certain concepts (for example, "harm" as a potential trigger for liability is closely related to the legal concept of "injury"), to avoid taking a normative position on torts-based versus statutory or regulatory approaches and to avoid prejudging the design questions presented here of how to impose legal disincentives for the sale of insecure software.

Second, to draw from the voluble historical debate and to help focus the current discussion onto a core set of design decisions and tradeoffs, this report surveys 123 academic articles and other pieces of writing that discuss some variation of software liability. These articles have been coded with respect to their stances on some of these design questions and examined both for trends in the balance of viewpoints as well as their evolution in time to seek to establish where there is existing consensus or relationships among variables that might inform the debate.

A note on scope: this paper is intended to address issues around liability for vendors of software related to cybersecurity practices and problems. Software liability as a term could encompass a wider range of potential considerations around software-mediated harms that could create legal liability, such as products-related liability for algorithmic systems. Legal liability can also arise for operators of software—such as liability for organizations that process personal data and experience a data breach—rather than the entity that created and sold such software. These questions are important but beyond the scope of this work.

[1] "National Cybersecurity Strategy," The White House, March 1, 2023, https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf.

[2] US Library of Congress, Congressional Research Service, *Introduction to Tort Law*, by Andreas Kuersten, 2023, IF11291.

Methods and Framework

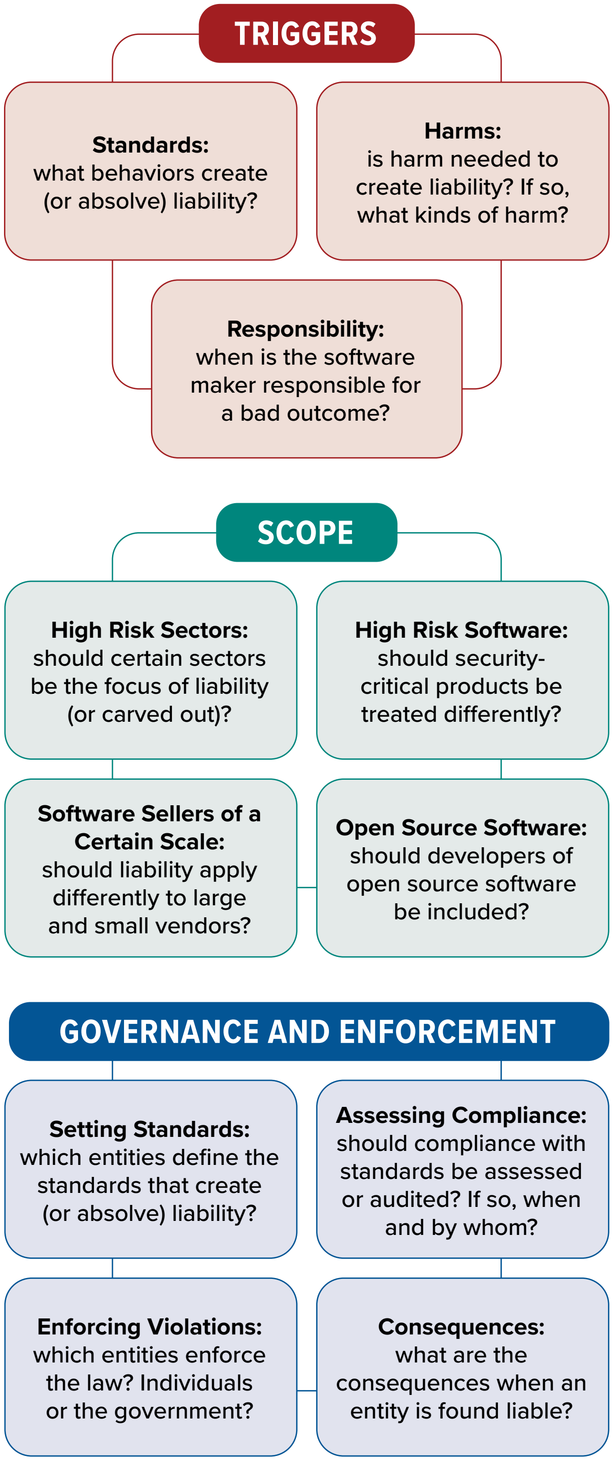
This report is based on a review of 123 pieces of writing from the scholarship on software liability, including law review articles as well as white papers, essays, and blog posts. The articles stretch over many decades of the debate – the earliest of the sampled articles was published in 1967, the most recent in 2023.

These articles were collected in two tranches: the first assembled by a single expert based on keyword searches of online scholarly databases; the second borrowed from a literature review created by an expert working group on the topic of software liability. This process resulted in a corpus of 171 articles, which was cut down during the coding process to a final corpus of 123 articles which were accessible and relevant to the topic of legal liability for insecure software.

Each of the collected articles was coded against a rubric developed by the authors to codify key policy design choices in the construction of a software liability regime. The articles were reviewed by two human coders who scored each article based on whether it endorsed or criticized the design choice or mentioned it without explicit criticism or endorsement. The threshold to distinguish between mention and endorsement or criticism was determined by the coders based on a holistic assessment of the viewpoint of the entire article, meaning there is necessarily an aspect of subjectivity in the data that appears below.

Due to the subjective nature of both the data collection and coding, the findings reported below should not be considered representative or statistically significant claims about the entire scholarly body of work relating to software liability. The analysis and visualizations are intended to illuminate certain broad trends and frame discussions of policy choices and models, a tool to inform the debate rather than an absolute claim about the state of consensus in a field.

In the literature sample, most articles focused on examining a specific component of or context for liability rather than a proposing a holistic regime, meaning that relatively few articles addressed every single aspect of this framework. For this reason, in many cases, visualizations address only those articles that address the question at hand in some form, while also seeking to contextualize how much of the broader sample of literature is included in that set.



TRIGGERS

Standards: what behaviors create (or absolve) liability?

Harms: is harm needed to create liability? If so, what kinds of harm?

Responsibility: when is the software maker responsible for a bad outcome?

SCOPE

High Risk Sectors: should certain sectors be the focus of liability (or carved out)?

High Risk Software: should security-critical products be treated differently?

Software Sellers of a Certain Scale: should liability apply differently to large and small vendors?

Open Source Software: should developers of open source software be included?

GOVERNANCE AND ENFORCEMENT

Setting Standards: which entities define the standards that create (or absolve) liability?

Assessing Compliance: should compliance with standards be assessed or audited? If so, when and by whom?

Enforcing Violations: which entities enforce the law? Individuals or the government?

Consequences: what are the consequences when an entity is found liable?

Triggers: What Makes You Liable?

Liability is typically understood as legal responsibility for one's actions (or inactions). From a policy perspective, what actions or inactions should make software makers legally accountable for poor software security?

There are two important concepts that are relevant across policy approaches: standards and harms.

Standards define good and bad behavior as it relates to developing secure software. Such measures range from design decisions such as choosing memory-safe programming languages or requiring user accounts to have multifactor authentication, to the use of tools or checks such as static analysis tools that scan code for vulnerabilities, descriptions of properties of code such as free of known vulnerabilities or known common weaknesses, or organizational practices such as having a security review step for code requests and secure release processes to avoid becoming a vector for a supply chain attack. The design of explicit standards, or decisions about how standards will implicitly be shaped over time, is a key part of a liability policy regime as it will define the behavior toward which software vendors are incentivized.

Harms relate to the ways in which insecurity can manifest itself in practice. Insecurity can manifest in code, such as in flawed code patterns that are vulnerable to prompt injections or that allow a user to bypass authentication, or in weaknesses in security-relevant processes such as code releases. Harms arise when such flaws are exploited to cause harm to the user of the software, from data breaches to ransomware, intellectual property theft, or physical injury.

Though regulatory liability could be triggered by a failure to meet standards alone, and torts are definitionally connected to a harm, both standards and harms play a role in each type of regime from a policy perspective. While not required in fact, in practice, enforcement for regulatory violations often follows news of a data breach or another harmful incident. For software torts, judges would need to consider questions that implicitly reply upon known or accepted standards or behavior with respect to cybersecurity, such as whether a software maker upheld a duty they owed to the user in creating the software (in negligence-based torts) or if the design they chose was foreseeably risky (under products liability).

Standards

A liability regime can take different approaches to defining the standards it includes and how it incorporates them. A law or regulation could reference frameworks or controls developed by standard-setting bodies such as the International Standards Organization (ISO) or the National Institute of Standards and Technology (NIST). A law could also create new standards through regulation, such as by directing an expert agency to create new rules. Alternatively, it could defer the question to the courts, by using a legal term left up to interpretation such as "reasonable cybersecurity measures." While explicit standards are more typical of a regulatory regime and case-by-case determination more typical for torts, articles and documents including the National Cybersecurity Strategy have endorsed hybrid models that combine torts with explicit standards in a "safe harbor" model, under which the law delineates a set of standards that, if a company can prove it upheld them, protect that company from tort liability. A safe harbor sets a behavioral "ceiling" for liability, dictating a level of behavior that wholly insulates entities from liability and thus defining an upper limit of the behavioral changes that a liability regime requires. Tort regimes could also use standards to define a "floor" on liability—a set of bad behaviors that create a presumption of negligence on the part of the software maker—while also leaving the door open for judges to examine specific cases and decide that software makers failed in their obligations to the software user in other ways.

Standards built explicitly into a regime, whether through regulatory approaches or a safe harbor in a tort regime, will delineate expected behavior by software makers more clearly and quickly than case-by-case approaches, which will need more settled cases (each of which can take years to resolve) to provide software makers with any measure of legal certainty about their obligations. On the other hand, avoiding a specific set of standards could make a regime more flexible, enabling a judge to review each case with respect to current industry best practices (which are always evolving, creating challenges for static regulation) as well as to use additional discretion to require safety behaviors that are above and beyond industry best practice[3].

This illustrates a general challenge in defining explicit standards: tradeoffs between flexibility and specificity. A simple

[3] "The T.J. Hooper," Casebriefs, accessed December 4, 2023, https://www.casebriefs.com/blog/law/torts/torts-keyed-to-epstein/the-negligence-issue/the-t-j-hooper-3/.

and specific list of practices that are easy for a company or an authority to audit for compliance may not be sufficient to guarantee that software is designed and implemented securely or to provide accountability for complex design flaws in software (see for example how businesses such as Microsoft, which espouse secure development principles,[4] have experienced severe incidents as the result of flawed design and implementation[5]), while standards that can encompass a wider class of design flaws provide less specificity and certainty for software makers. For example, concepts such as "secure-by-design" and "secure-by-default" as recently championed by the Cybersecurity and Infrastructure Security Agency (CISA)[6] are powerful principles that span multiple levels of abstraction from principles to specific practices. However, the highest level and most encompassing principles from this framework may be challenging to define in a way that makes it easy for businesses to ensure their compliance or for a potential enforcer to easily prove noncompliance.

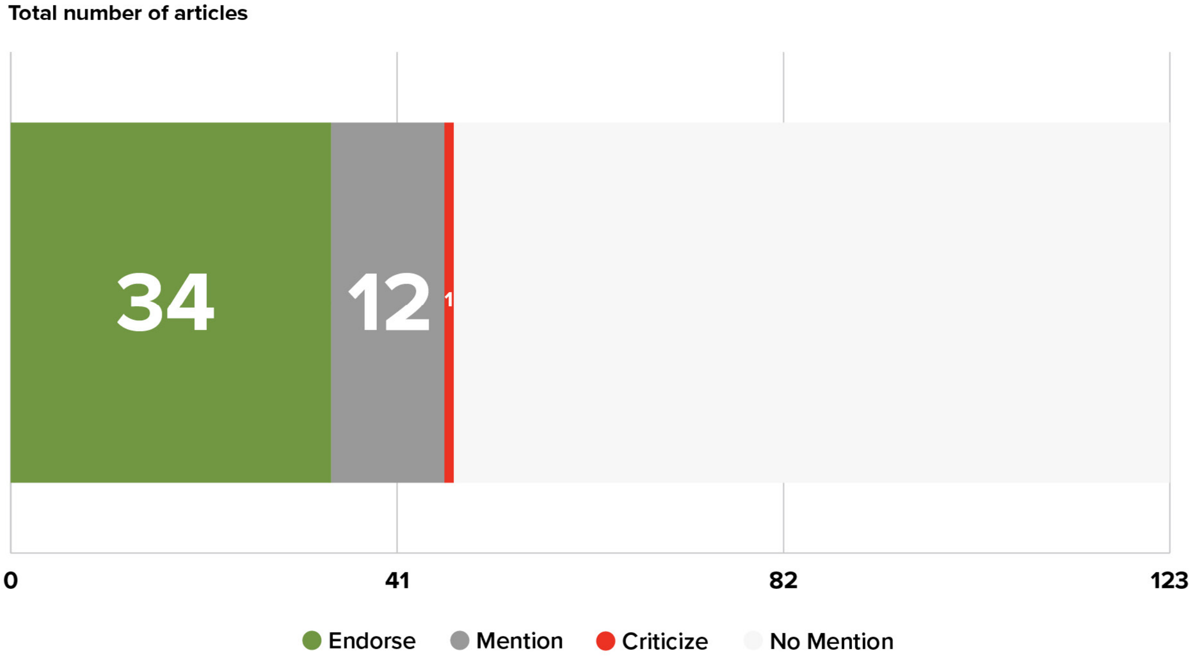
47 of the 123 articles surveyed mentioned the idea of using secure development standards as a basis for standards in a liability regime, with 34 of those articles explicitly endorsing secure development standards as a component of a liability regime.

Such standards appear to have been relatively popular over time within the sampled literature, having been mentioned since the late 1980s.

Some of these articles mentioned only the general idea of incorporating such secure development standards into a regime or suggested entities that could develop such standards, while others named specific standards, including government-developed standards such as NIST's Secure Software Development Framework (SSDF) or those developed by standards organizations such as the International Organization for Standardization (ISO) or the Institute of Electrical and Electronics Engineers.

NIST's SSDF is a framework created to "reduce the number of vulnerabilities in released software, reduce the potential impact of the exploitation of undetected or unaddressed vulnerabilities, and address the root causes of vulnerabilities to prevent recurrences." It includes suggestions to

Standard: Using Secure Development Practices



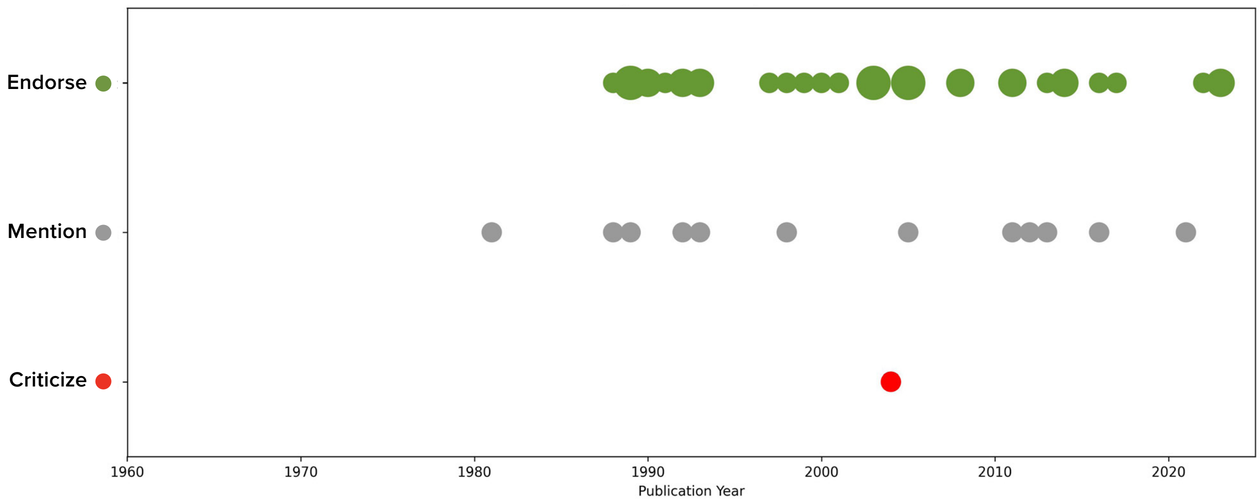
Source: Cyber Statecraft Initiative

[4] "What are the Microsoft SDL Practices," Microsoft, accessed December 4, 2023, https://www.microsoft.com/en-us/securityengineering/sdl/practices.

[5] Dan Goodin, "Microsoft Finally Explains Cause of Azure Breach: An Engineer's Account Was Hacked," *Ars Technica*, September 6, 2023, https://arstechnica.com/security/2023/09/hack-of-a-microsoft-corporate-account-led-to-azure-breach-by-chinese-hackers/.

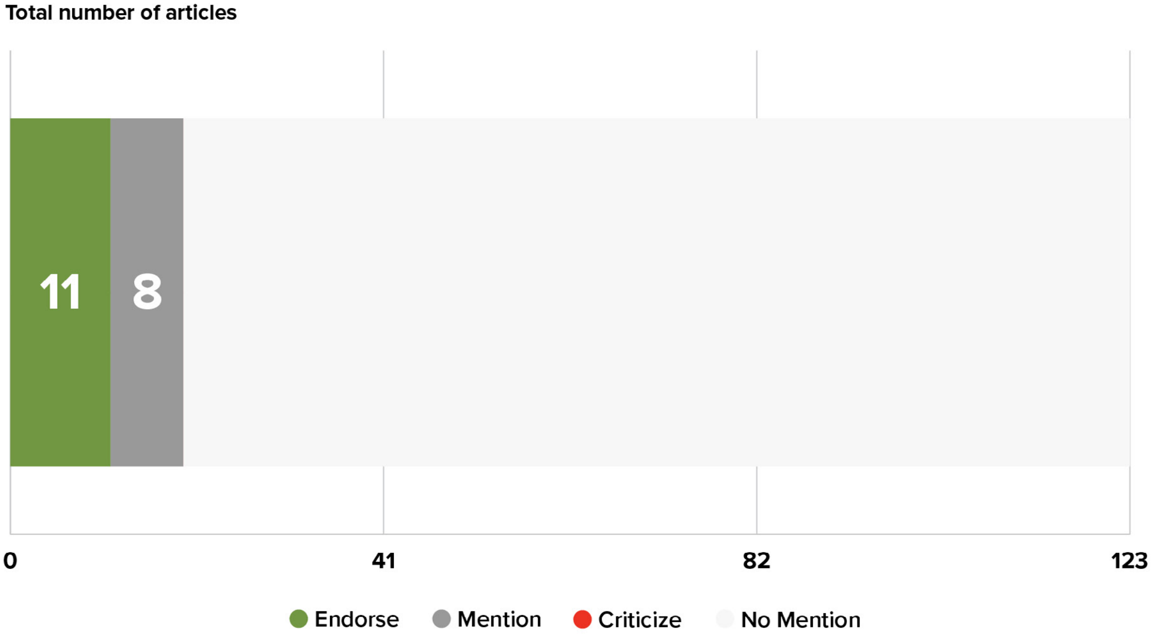
[6] "Secure by Design," Cybersecurity and Infrastructure Security Agency (CISA), accessed December 4, 2023, https://www.cisa.gov/securebydesign.

Standard: Using Secure Development Practices



prepare an organization (such as developing organizational policy with respect to software security procedures), to protect software (such as using version control and code and commit signing), to produce secure software (such as using risk modelling, documenting design decisions, performing human or software-based security auditing, evaluating third-party software components), to follow secure coding practices (such as avoiding unsafe functions or unverified inputs, and selecting secure default configurations), and to respond to vulnerabilities (such as gathering and investigating reports, planning and implementing risk-based remediations, and analyzing root causes to

Standard: Disclosing and Remediating Vulnerabilities



Source: Cyber Statecraft Initiative

feed back into security processes).[7] The NIST standards combine elements that speak to the security of the code itself with those that address an organization's relevant policies, the security of their development processes as a potential vector for supply chain attacks, and their behavior with respect to known good practices such as addressing vulnerabilities and performing security audits. By executive order, the US government has moved toward requiring its software vendors to comply with the NIST SSDF; CISA has instantiated requirements based on the SSDF into a secure software self-attestation form that (once finalized) will need to be completed by all vendors who sell software to the government.[8]

The coding rubric also included a few specific elements of such frameworks to see how often they were specifically named in the articles. Many fewer articles—only 19 of 123—focused on requirements for software makers to have policies, procedures, or specific behaviors with respect to how they address or disclose vulnerabilities in their code, and only a single article explicitly discussed code security auditing or penetration testing as a part of a regime.[9]

Harms

A liability regime may or may not require, for liability to accrue, that software insecurity causes actual harm to software users. Regimes based on torts almost definitionally require a harm to trigger liability, but regulatory regimes can simply require certain behavior of software makers.

One disadvantage of requiring harm to trigger software liability is that cyber outcomes (and thus harms) are dependent not only on the actions of the software maker, but also on the actions of an adversary or bad actor that exploits a vulnerability to cause harm. This adds into the equation complicating questions about the skills and capabilities of different kinds of adversaries and whether it is fair or desirable to hold software makers equally responsible if they are hacked by a sophisticated and well-resourced entity such as a nation-state, versus by run-of-the-mill cyber criminals. On the other hand, hinging liability on harms, in a sense, scales enforcement to the manifested negative consequences of insecurity, providing an inbuilt mechanism for imposing harsher punishments on those entities whose insecurity is more societally deleterious or costly.

Harms from cyber incidents can include costs to businesses, negative consequences for individuals such as the loss of privacy, and harms to national security such as through the theft of intelligence-relevant information. Financial costs to businesses are perhaps the best understood and best-represented under existing theories of tort liability (with some major caveats to be addressed later). Businesses impacted by a cyber incident can face financial costs stemming from operational disruptions or data loss; ransomware payments; technical remediation and incident response; notifying impacted consumers and providing identity monitoring; declines in share prices; and fines or lawsuits from government or shareholders.. Estimates of the precise costs of cyber incidents vary widely, but CISA reported several studies with estimates for the median cost of an incident ranging between $50,000 and $250,000 and the mean ranging between $400,000 and $7 million.[10]

Albeit less common than financial harms, cyber incidents can also cause physical harm. Physical harms from cyber incidents are likelier to arise from high-stakes, software-enabled products such as medical devices, airplanes, and cars.

Questions around which types of harms can create liability for software makers were widely discussed in the liability literature surveyed, perhaps in part because such questions have frustrated past attempts to use common law torts to bring cases against the makers of insecure software. "Economic loss doctrine," a legal theory in place in many states, holds that product liability should not allow one party to seek compensation for economic damages—essentially, any harms outside of physical harms or property damage—beyond what was outlined in the contract they agreed to.[11] Because software often causes only financial harms to impacted businesses, and because software vendors often sell or license software under contracts that absolve them of most liability, this doctrine has limited the success of past tort cases for software insecurity.

Discussed in 89 articles, the question of which harms can potentially trigger liability was one of the most-discussed

[7] Murugiah Souppaya, Karen Scarfone, and Donna Dodson. "Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities." National Institute of Standards and Technology US Department of Commerce, February 2022. https://doi.org/10.6028/NIST.SP.800-218.

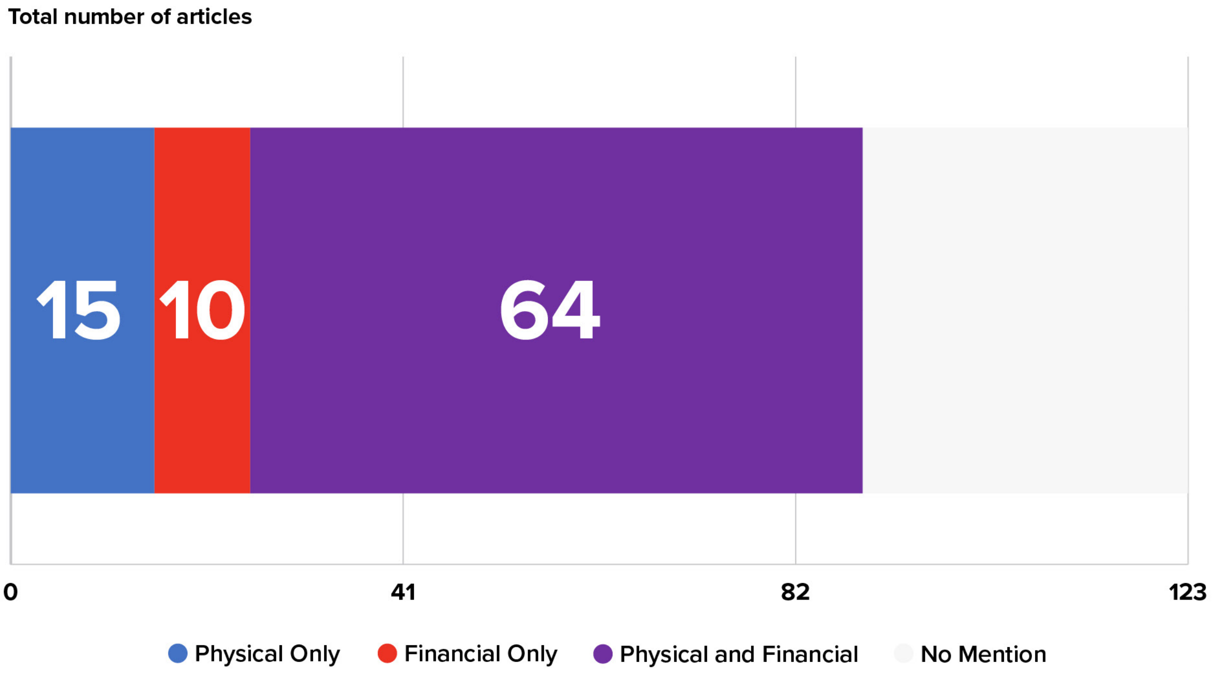
[8] "Request for Comment on Secure Software Development Attestation Common Form," Cybersecurity and Infrastructure Security Agency (CISA), accessed January 5, 2024, https://www.cisa.gov/secure-software-attestation-form.

[9] Jane Chong, "Bad Code: The Whole Series," *Lawfare*, November 4, 2013, https://www.lawfaremedia.org/article/bad-code-whole-series.

[10] "Principles and Approaches for Secure By Design Software, 2023, https://www.cisa.gov/sites/default/files/publications/CISA-OCE_Cost_of_Cyber_Incidents_Study-FINAL_508.pdf.

[11] Catherine M. Sharkey, "Can Data Breach Claims Survive the Economic Loss Rule?," *DePaul Law Review* 66 (2017), last updated August 21, 2017, https://ssrn.com/abstract=3013642.

Harm: Types of Harm that Can Trigger Liability



Source: Cyber Statecraft Initiative

in the literature, behind only the questions of product and negligence-based torts. In papers that explicitly mentioned the question of which types of harms should qualify, the majority view was that both economic and physical harms should serve as a potential basis for liability.

Responsibility

A liability policy regime will also need to consider how to allocate responsibility for failures between software manufacturers and software users. Software security is a problem of "shared responsibility": users of software, in addition to its developers, have significant control over cybersecurity outcomes through their own security practices. Torts already have conceptions of "comparative negligence" when the behavior of the harmed party contributed significantly to the harmful outcome—policymakers might want to map this concept explicitly to the software context to balance certain policy goals.

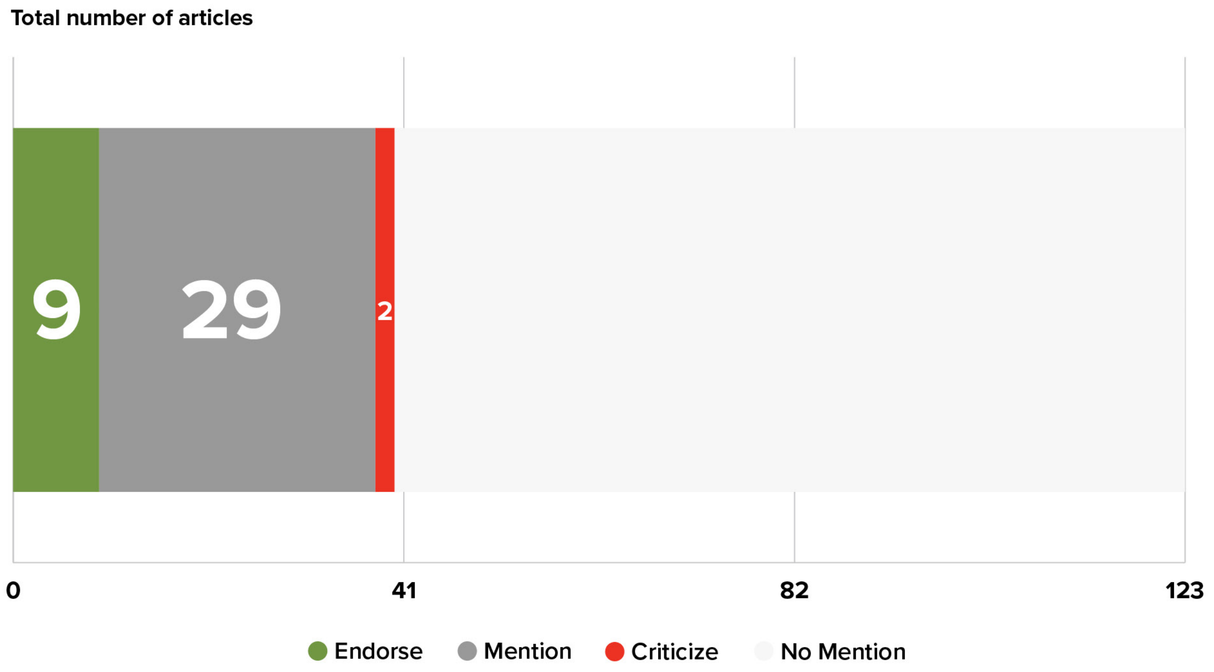
The most canonical question around the allocation of responsibility in software liability regimes is around "patching," the practice in which vendors release fixes for discovered vulnerabilities and bugs in the form of software updates that their customers must then apply. Put simply, should a vendor continue to be liable for harms arising from a vulnerability, even after they released a patch that would fix it (and the customer failed to apply it)?

On the one hand, frequent patching is an ongoing challenge for many organizations,[12] especially those with the least resources to dedicate to information technology management and security.[13] A world in which vendors ship insecure code and then inundate their users with countless security-critical patches seems undesirable, and holding developers liable for code regardless of patch availability would certainly incentivize them to release more secure code. At the same time, expecting developers to release fully and perpetually secure software is likely an unrealistic goal, and patching is thus a relatively accepted part of the current software delivery paradigm. There exist genuine policy goals both in reducing the number of patches that organizations need apply, and in providing incentives for software developers to release patches in a timely fashion

[12] "2022 Top Routinely Exploited Vulnerabilities," Cybersecurity and Infrastructure Security Agency (CISA), August 3, 2023, https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-215a.

[13] Evan Sweeney, "For Hospitals Defending against Cyberattacks, Patch Management Remains a Struggle," *Fierce Healthcare*, May 17, 2017, https://www.fiercehealthcare.com/privacy-security/for-hospitals-defending-against-cyberattacks-patch-management-remains-a-struggle.

Standard: Requirements for Software Users (e.g. Timely Patching)



Source: Cyber Statecraft Initiative

and for software users to apply these patches. Any liability regime that rests on or can be triggered by harm will need to draw lines in the sand about whether and when, once a vulnerability is known and a patch available, subsequent bad outcomes are the fault of the developer or the user.

Beyond just timely updating, there are other practices in the security context that software operators control that contribute significantly to security outcomes.[14] Software operators must maintain firewalls and monitoring capabilities on their network. They must correctly configure products and choose secure settings. If a software liability regime seeks to incorporate some concept of comparative negligence for cases in which the software operator's actions (or inactions) contributed significantly to the harm that arose from the software's insecurity, it may also need—explicitly or implicitly—standards for the behavior of software operators and developers.

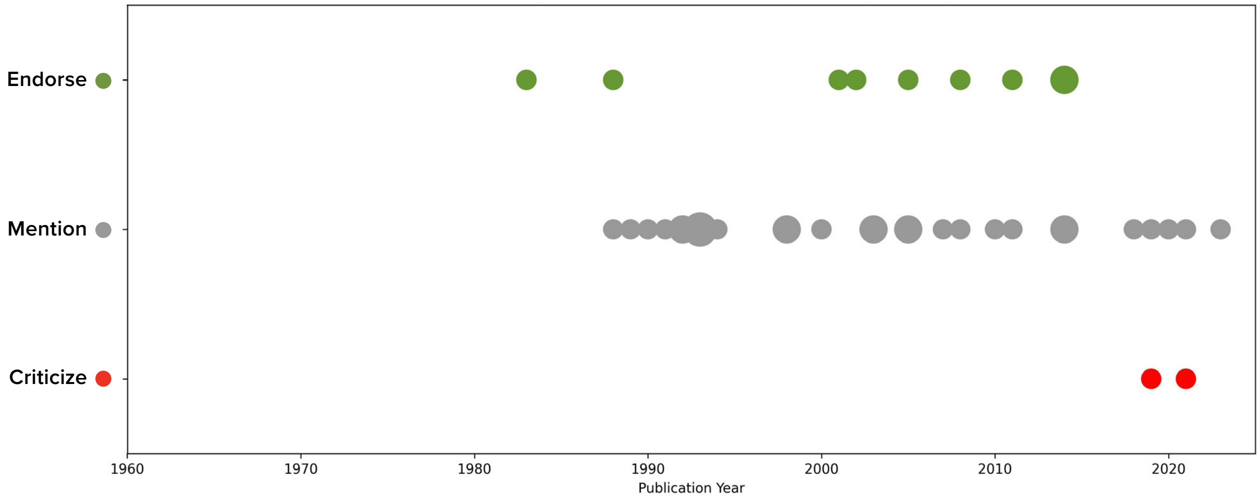
40 of the articles surveyed mentioned the idea that a liability regime for software makers should codify considerations or requirements pertaining to the behavior of the software user, such as questions about whether a software patch was available but unapplied. Nine of those articles explicitly endorsed the idea and two critiqued it, with these two critiques occurring more recently than any of the endorsements.

Some articles from the literature examined other potential policy approaches to the patching problem such as "patch liability," the idea of instead requiring software developers to pay the costs associated with the resources their customers need to expend in order to apply software patches.[15] In general, these ideas appear relatively underexplored relative to the complexity of the policy tradeoffs at play, with only a few articles mentioning the potential impacts of different liability approaches on developers' and users' incentives and behaviors with respect to patching.

[14] "2022 Top Routinely Exploited Vulnerabilities."

[15] Terrence August and Tunay I. Tunca, "Who Should Be Responsible for Software Security? A Comparative Analysis of Liability Policies in Network Environments." *Management Science* 57 (2011): 934–59, http://www.jstor.org/stable/25835749.

Standard: Requirements for Software Users (e.g. Timely Patching)



Scope: Who Can be Liable?

Scope describes the myriad questions around which software and software vendors fall under the purview of a liability regime.

Software for High-Risk Sectors

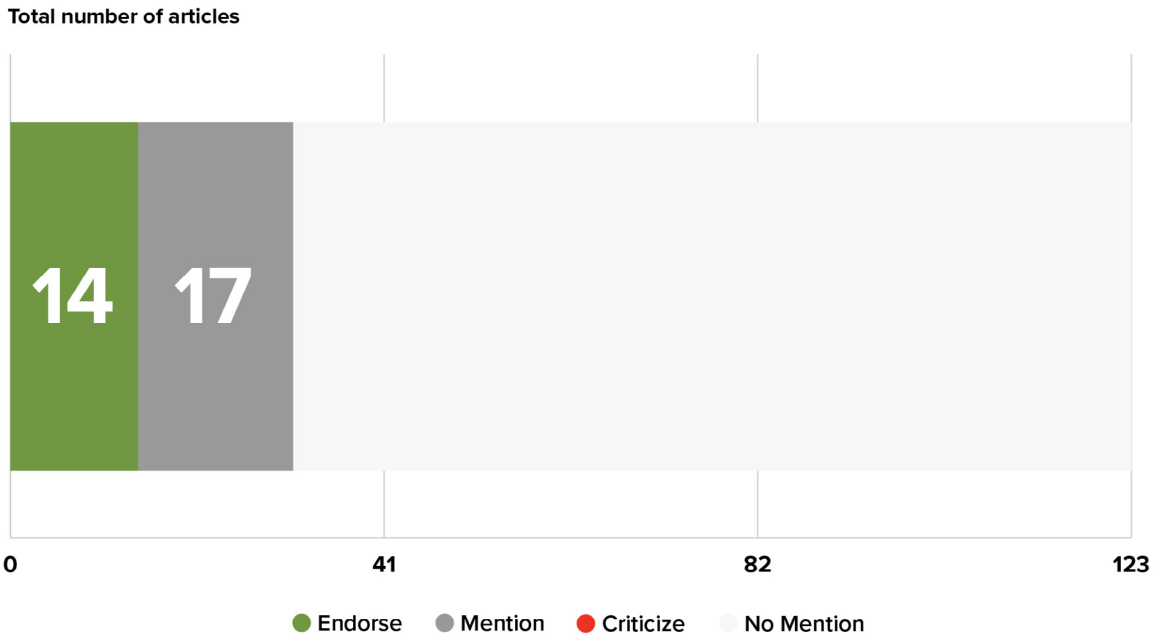
One way to scope a liability regime would be to limit its requirements to a specific sector or application in which software might operate (or to include multiple sectors but to tailor elements such as standards to each). It makes certain intuitive sense from a policy perspective to apply higher standards of cybersecurity care for manufacturers of medical device or airplane software than creators of general-purpose word processing or customer management software. This approach would generally mirror the approach taken with existing cybersecurity standards for software operators in the United States, which tend to apply for specific high-risk sectors or data processing activities.

Within the literature, 31 articles explicitly discussed considerations around sector-specific scoping or sector-specific standards for software liability. Just under half of the articles which mentioned the idea endorsed it, and both endorsements and neutral mentions stretch over multiple decades of the debate.

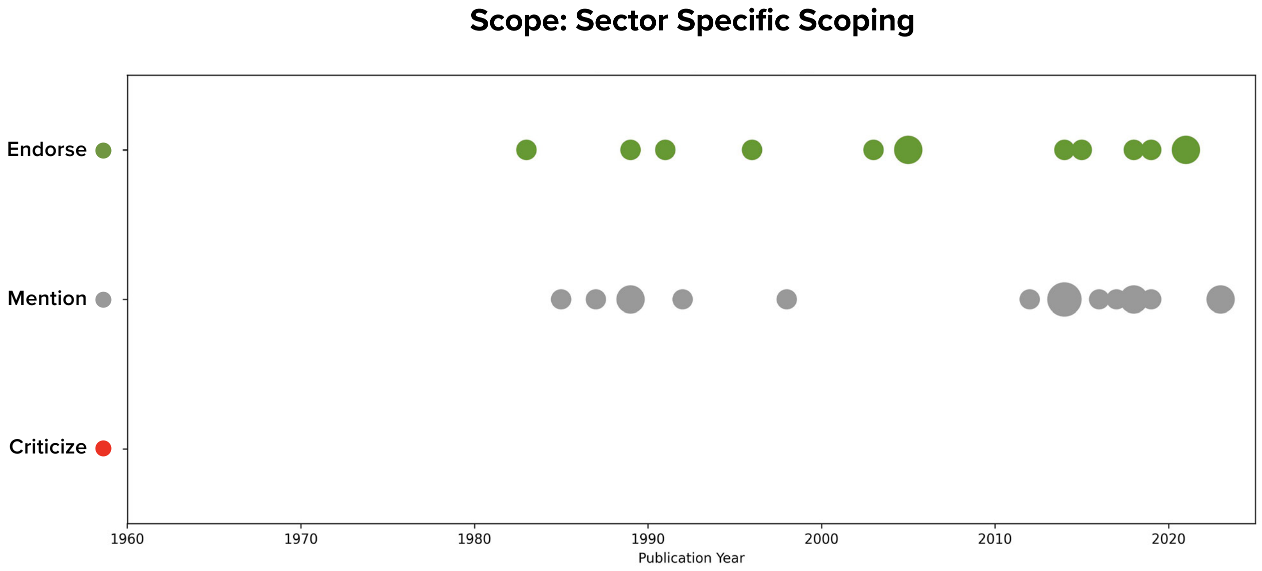
Within the literature, healthcare and medical devices were most often mentioned as sectors that might be treated differently, with articles also mentioning autonomous vehicles, airplanes, voting machines, and nuclear plants. These sectors typical combine both potentially unique, application-specific software such as software embedded into medical devices, airplanes, or voting machines with heightened risks of potential bad outcomes (often but not always in terms of potential loss of life) from insecurity.

A liability regime could adopt a model premised on specific kinds of sector-specific software (e.g., heightened liability for makers of autonomous vehicle software) or one premised on liability for any type of software used by high-risk sectors (e.g. heightened liability for any type of software sold to autonomous vehicle companies). The latter model faces a challenge in the fact that many types of software can be used across high- and low-risk sectors without distinction by the vendor or due notice by the customer. Many types of software are purpose-general (e.g., email clients) and can be deployed across a broad range of organizations and operating contexts, creating a leveling problem between the design context of software and its use. Cabining liability to certain types of software that are specific and high risk within these sectors appealingly avoids this problem. Yet,

Scope: Sector-Specific Scoping



Source: Cyber Statecraft Initiative



clear line drawing is a problem even under this approach, with many examples of technologies that provide essential support to the function of such devices but that are not specific to them, such as operating systems or cloud data processing. Depending on how these lines are drawn, software for use in these sectors is likely to become more expensive and more bifurcated than standard, consumer-grade applications.

Software for High-Risk Functions

Some types of software are sensitive not because of the context in which they are deployed and used, but instead because they perform security-critical or risky functions. For example, identity and access management systems control access to other computing resources and are frequently targeted by hackers seeking to escalate their permissions to access sensitive data or perform privileged actions. Other software systems with potentially important and systemic security impacts include tools like hypervisors and virtualization software in cloud computing environments or network management tools and firewalls. Different applicability or standards for software of different security risk levels are present in existing policy regimes such as the European Union's Cyber Resilience Act, which makes use of such a distinction and applies higher standards of security to software performing certain high-risk and security-critical functions.[16]

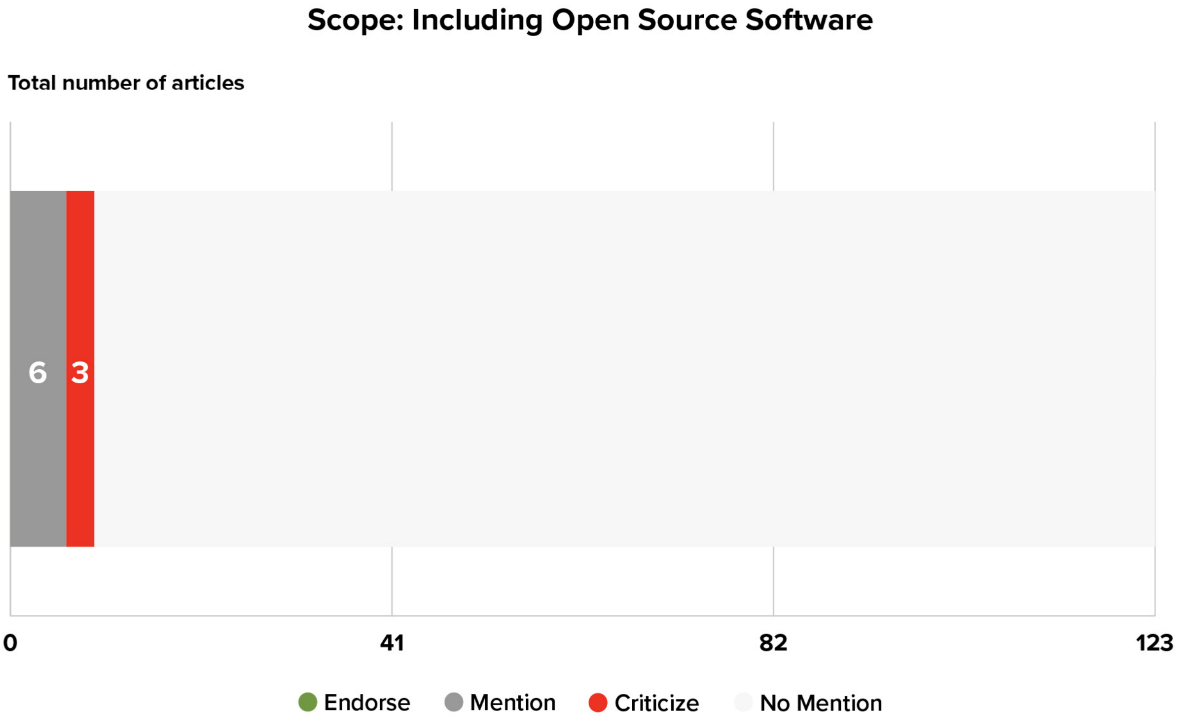
Software Sellers of a Certain Scale

Another standard that a law could use to scope who can be liable—or to tier other elements of the regime, such as standards—would be based on the size of the entity that sold the software. For example, liability could kick in once companies are of a certain size as defined by financial metrics such as revenue, or sales of the software in question (noting that this question might be difficult to answer—for example, how to treat the sale of one license to one company, but that may result in hundreds of installs of the software). Conversely, small entities—those with low revenues or that have sold few instances of the software in question—could be carved out of a liability regime or subject to less complex or burdensome security standards. Such differentiation would reduce the compliance burden for small businesses that sell software. Such a system could also intersect with other scoping or tiering systems; for example, it might be the case that a software vendor that sells software to a water treatment plant or power station should always be liable, regardless of size, while the same might not be true for those that sell to non-critical infrastructure companies.

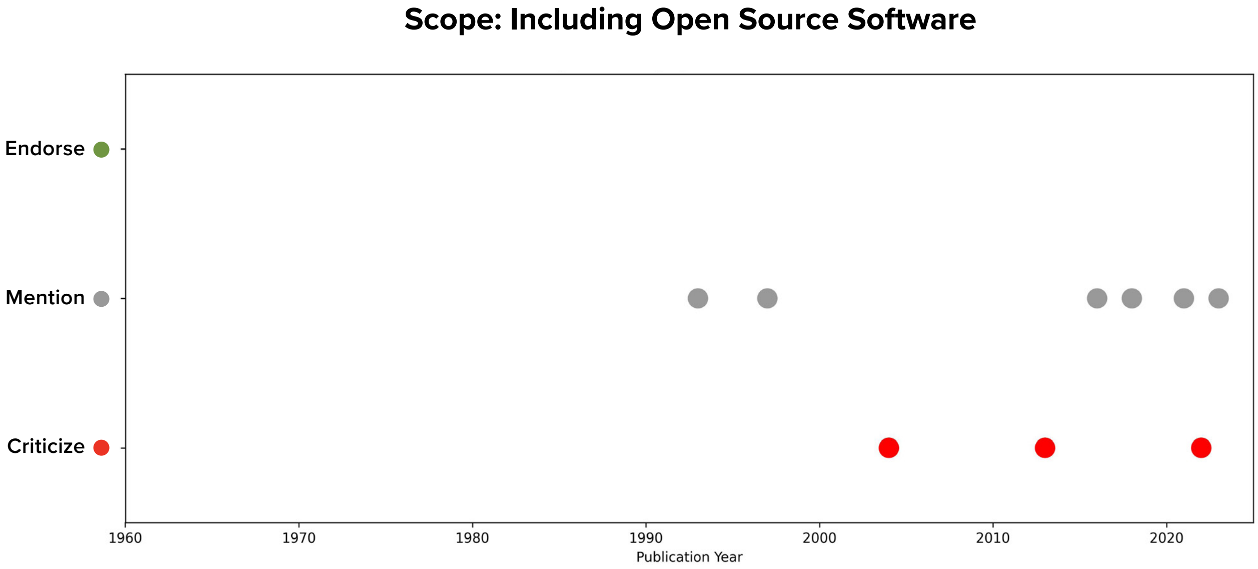
Open Source Software

Open source software (OSS) is not software sold by a vendor; rather, it is software whose source code is publicly available, distributed under a license that grants others total

[16] Markus Limacher, "Cyber Resilience Act – Get Yourself and Your Products up to Speed for the CRA," *InfoGuard*, December 4, 2023. https://www.infoguard.ch/en/blog/cyber-resilience-act-get-yourself-and-your-products-up-to-speed-for-the-cra.



Source: Cyber Statecraft Initiative



permission to use and modify the software, while ensuring that the software's original creator offers no guarantees about its use nor accepts responsibility for any harms caused.

While discussion of OSS has increased since the year 2010 as compared to prior decades, it has only been mentioned in 9 articles, much less than many of the other design questions in the framework. Of all articles that mentioned the question, none endorsed the idea of including developers of OSS in a software liability regime.

This finding broadly aligns with the authors' prior supposition that liability is not the right policy tool to use to improve security in the open source ecosystem. Much of open source code is often published by academics, researchers, and hobbyists; threatening these unpaid volunteers with legal liability for sharing their code would likely have a chilling effect on their participation and thus harm an ecosystem that has provided myriad benefits for academic knowledge-sharing and the distribution of useful components. Even for widely used and widely supported open source packages, creating potential liability for contributors could disincentivize hobbyists and corporate employees alike from contributing security features and fixes back to the package—exactly the opposite of what most open source packages need from a security perspective. Besides these issues, there are more practical ones, such as to which contributors to apply liability when open source packages often incorporate contributions from dozens or hundreds of developers. There are myriad other ways to support the security of OSS (funding, auditing, encouraging companies to contribute back to OSS security[17]) that are a better fit for the unique context of open source that lacks clear contracts, transactions, or payments between a software's developer and its users.

The inclusion of open source developers is not the only means by which a liability policy regime could interact with open source software security. A liability regime could place requirements around responsible use of open source code on software vendors as an element of standards. This would incentivize software vendors to more carefully vet and to contribute back to the security of open source code that they want to use, improving the health of the broader ecosystem and the security of proprietary code that incorporates open source while avoiding the chilling effects of placing liability directly onto the developers of open source code.

[17] Stewart Scott, Sara Ann Brackett, Trey Herr, Maia Hamin, "Avoiding the Success Trap: Toward Policy for Open-Source Software as Infrastructure." *Atlantic Council* (blog), February 8, 2023, https://www.atlanticcouncil.org/in-depth-research-reports/report/open-source-software-as-infrastructure/.

Governance and Enforcement: Who Holds You Liable (and How)?

Equally important to the "what" of a liability regime is the "who." That is, which entities are responsible for implementing the components that make up the regime? Enforcement and governance are essential elements that differentiate liability from mechanisms of self-governance or voluntary standards.

Setting Standards

There are a few existing models of standard-setting that might be ported over to the software cyber liability context.

One model would be akin to that taken by certain cyber-physical systems such as airplanes and medical devices: in this model, a regulator sets standards for disclosures or information that a software product must submit to the regulator before the product comes to market. For example, the Federal Aviation Administration has increasingly embedded cybersecurity into its approval processes for airplanes,[18] and the Food and Drug Administration (FDA) requires medical device makers to adopt and disclose standards around secure development before their devices can be approved to go to market.[19] These preemptive approval models allow regulators to more easily include standards around secure development into their processes: rather than needing to give companies a checklist up-front of practices by which they must abide, they can force companies to affirmatively attest to or describe the secure-by-design and secure-by-default practices they followed in the creation of their software. In these models, the same entity also certifies compliance (e.g., allows the product to come to market) and, often enforces against violators (although in medical devices, for example, consumers can also bring suit under products liability). These models are relatively powerful, but they hinge on the fact that the regulator controls entry to the market, in that their approval is required as a precondition of the product being sold. This model is less realistic for all software products—software ranges from industrial control systems to video games created by small independent developers, and requiring even the smallest of software programs to be approved before coming to market would likely result in a severely throttled software ecosystem.

Another model would be having an expert agency set standards such as secure development standards, which would apply to certain types of software without requiring disclosures or filings before a product comes to market. In most models from existing law, the entity that sets the standards is also the one that enforces them [e.g., the Federal Trade Commission (FTC) both sets standards for and enforces the Gramm Leach Bliley Act,[20]], but sometimes the two functions are divided. Yet another model would be requiring software makers to include statements of compliance with particular (federally selected or developed) standards in their contracts, thus giving software buyers the opportunity to sue software vendors for contractual violations if they fell short. For example, a recent proposed update to the Federal Acquisition Regulation would require contractors developing software on behalf of the government to certify compliance with Federal Information Processing Standards developed by the National Institute of Standards and Technology (NIST); if a company misrepresents their compliance, an action can brought by the Department of Justice under the False Claims Act.[21]

Another approach would be to avoid prespecification of standards altogether. For example, a law could state that a company has a duty to its customers to uphold "reasonable" security standards, thereby allowing a judge in a case to determine what measures are reasonable. In such cases a judge may well look to existing standards and industry best practices to judge whether a practice was or was not reasonable—but these standards and practices are not identified a priori in the regime itself. As discussed in the section on standards, this approach create flexibility by trading off speed and certainty.

[18] "Advisory Circular on Guidelines for Design Approval of Aircraft Data Link Communication Systems Supporting Air Traffic Services (ATS)," US Department of Transportation, Federal Aviation Administration, September 28 2016, https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_20-140C.pdf.

[19] "Cybersecurity in Medical Devices: Quality System Considerations and Content of Premarket Submissions," US Food and Drug Administration, Center for Devices and Radiological Health, September 26, 2023, https://www.fda.gov/regulatory-information/search-fda-guidance-documents/cybersecurity-medical-devices-quality-system-considerations-and-content-premarket-submissions.

[20] "Gramm-Leach-Bliley Act," Federal Trade Commission, June 16, 2023, https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act.

[21] "Government Contractors Beware: New Cybersecurity Rules and False Claims Act Enforcement Actions on the Rise," Akin Gump Strauss Hauer & Feld LLP, accessed December 4, 2023, https://www.akingump.com/en/insights/alerts/government-contractors-beware-new-cybersecurity-rules-and-false-claims-act-enforcement-actions-on-the-rise.

Assessing Compliance

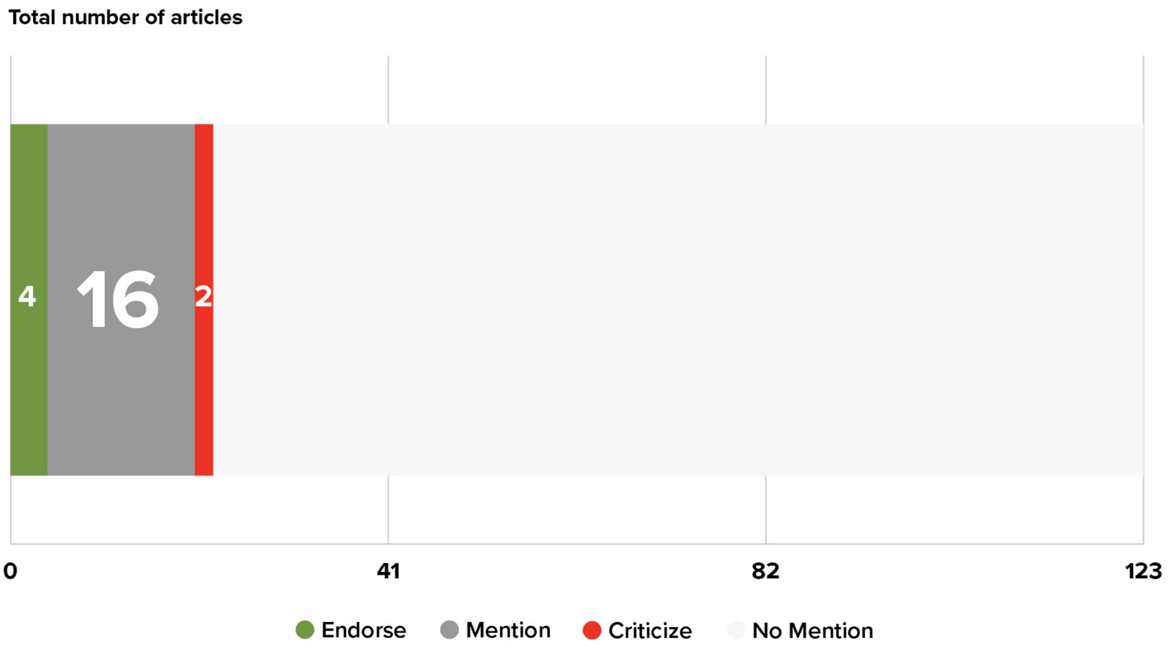
Depending on the structure of a liability regime, some entity or entities may be empowered to audit, assess, or certify compliance as part of the scheme. One approach would be self-certification—requiring entities to certify their own behaviors or compliance with standards, facing penalties if their attestations were later found to be false. Self-certification would likely need to be paired with some requirements for what entities must certify, to avoid race-to-the-bottom situations in which companies seek to promise nothing so they can be accountable for nothing. Self-certification was mentioned in only four of the articles and endorsed by none. However, it is a component of existing regimes such as Europe's Cyber Resilience Act.

Other approaches would involve external auditing of some form. External auditing to determine compliance was mentioned by relatively few articles, which were split on its desirability.

External auditing could take several forms. A regulator could certify compliance as a prerequirement for the sale of software—mirroring regimes such as the approval processes for medical devices and airplanes outlined above, or the Federal Motor Vehicle Safety Standards.[22] Alternatively, audits could be reactive rather than proactive, such as those performed by the Health and Human Services (HHS) Office of Civil Rights to investigate inbound tips and assess compliance.[23]

External auditors could also come from outside government; government can certify outside entities to assess compliance with the standards of the regime. For example, the Children's Online Privacy Protection Act allowed industry groups to certify self-regulatory frameworks, which, after approval by the government, satisfy the law's safe harbor requirements.[24] Liability regimes can also combine other variables such as scope with auditing requirements: the European Union's (EU) Cyber Resilience Act allows noncritical entities to perform a self-evaluation of their

Governance and Enforcement: Requiring External Auditing



Source: Cyber Statecraft Initiative

[22] "Laws & Regulations," NHTSA, accessed December 4, 2023. https://www.nhtsa.gov/laws-regulations.

[23] "Enforcement Process," US Department of Health and Human Services, May 7, 2008, https://www.hhs.gov/hipaa/for-professionals/compliance-enforcement/enforcement-process/index.html.

[24] "COPPA Safe Harbor Program," Federal Trade Commission, January 7, 2015, https://www.ftc.gov/enforcement/coppa-safe-harbor-program.

conformity with the requirements of the Act, while critical entities must be certified by an external (EU-approved) auditor.

Enforcing Violations

One general dividing question is, do companies and users have the right to directly sue those responsible for insecure software, or does a government entity (e.g. a federal agency such as the FTC or State Attorneys General) enforce the law? Although the two delineate general models for enforcement, they are not mutually exclusive.

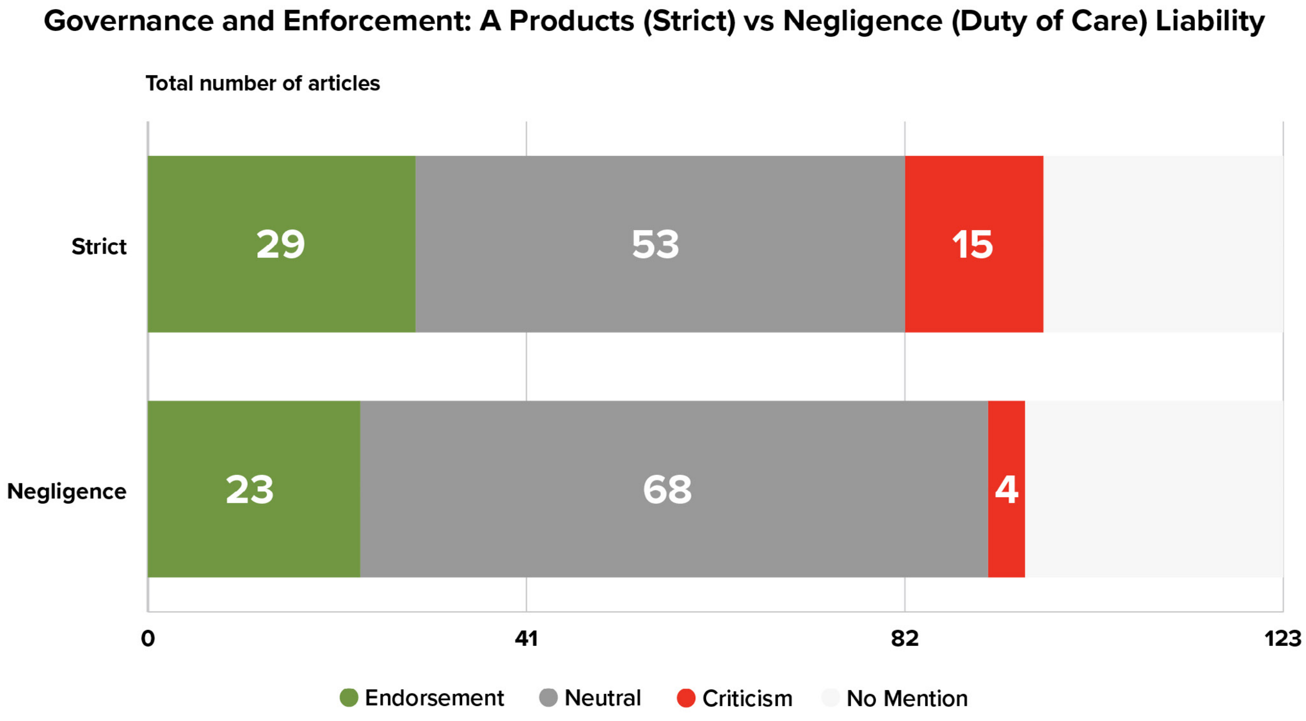
CONSUMER ENFORCEMENT

One option for enforcement is to allow the entities harmed by insecure software—perhaps most often businesses, but also including individual consumers—to directly sue the company that sold them the software. Such a regime could be brought about by passing a law to change how product or negligence torts have been interpreted by the courts when it comes to software insecurity. Alternately, a law could simply establish new responsibilities or obligations that software makers owe to their customers and include a private right of action that allows those customers to directly sue software makers that have violated their rights under that law.

Product or negligence torts for software were the two most widely discussed topics coded in the literature, with 97 and 95 mentions of each concept respectively. Data on article stances shows that product liability has both more supporters and more detractors than does a negligence standard. Generally speaking, authors adopted an either/or approach: of the 86 articles that mentioned both concepts, only seven articles endorsed both approaches.

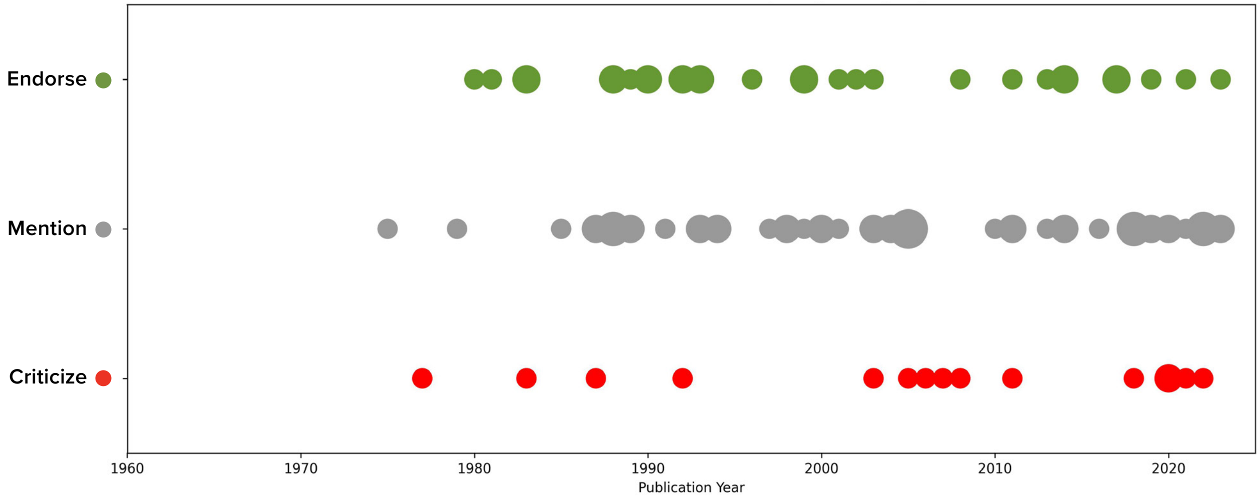
Visualizing the distribution of these articles by their year of publication suggests that this debate has been ongoing since the beginning of the literature sample and that neither approach has come to dominate over time.

Another approach that Congress could take to structure a law with consumer enforcement would be a private right of action, a federal law that places obligations on companies and then grants consumers the right to bring suit to enforce their rights under the law. Eight articles endorsed the idea of allowing both government and consumer enforcement by creating a federally enforced regime with a private right of action.

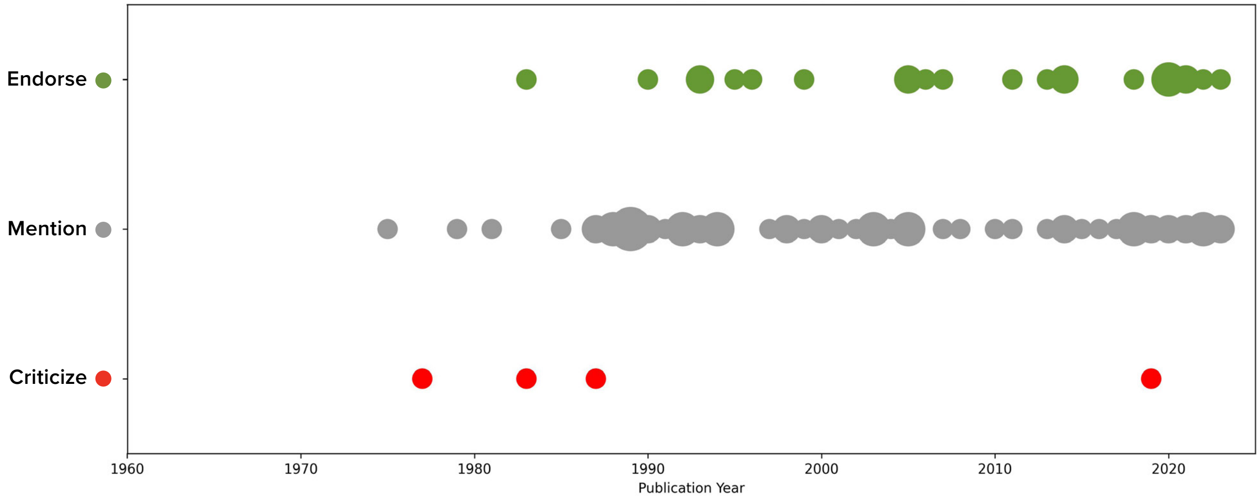


Source: Cyber Statecraft Initiative

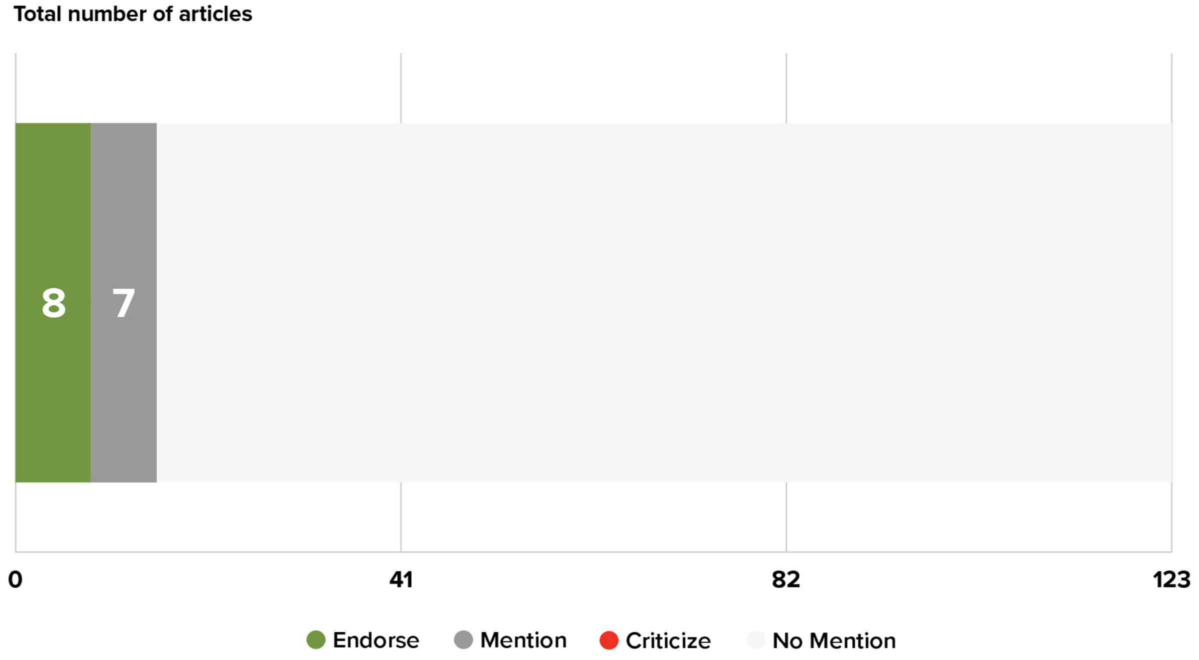
Governance and Enforcement: A Products (Strict) Liability Regime



Governance and Enforcement: A Negligence Liability (Duty of Care) Regime



Governance and Enforcement: A Federally Enforced Regime with a Private Right of Action



Source: Cyber Statecraft Initiative

GOVERNMENT ENFORCEMENT

Another approach, and one taken with many existing cyber standards, is to have a federal or state agency (or agencies) enforce the law's requirements instead.

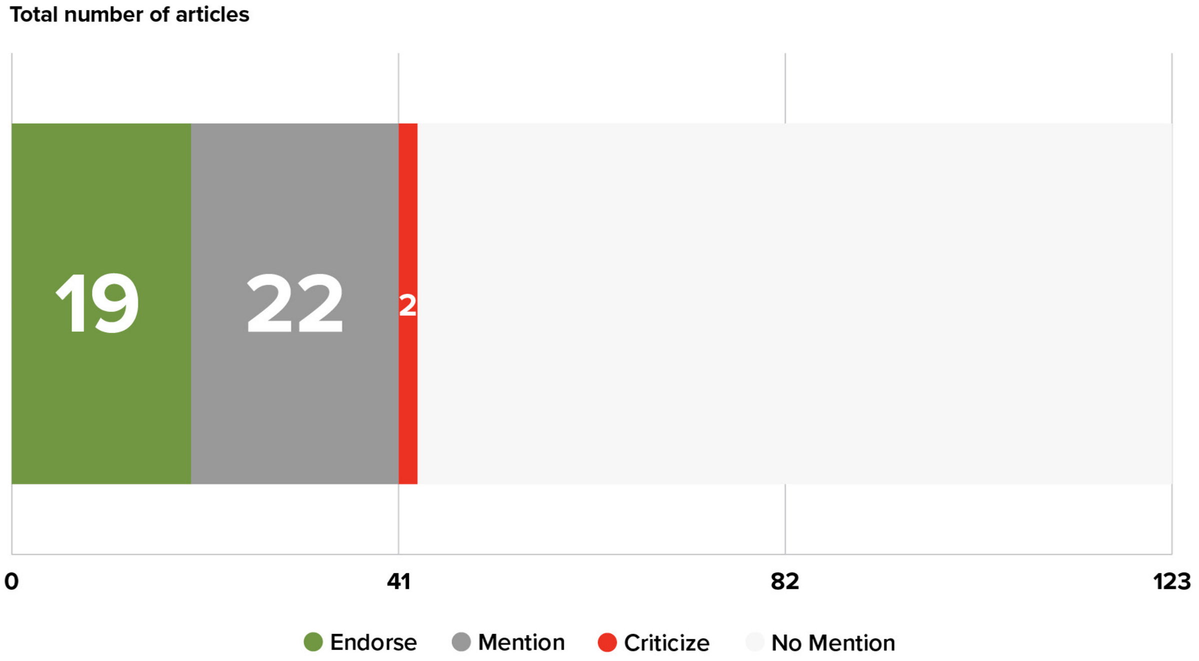
Many existing federal-level cybersecurity standards in the United States are sector-specific and thus enforced by the sector regulator, or by the Federal Trade Commission (FTC) if no such is available: for example, Health Insurance Portability and Accountability Act (HIPAA), the law imposing cybersecurity standards on healthcare entities in the processing of health data, is enforced by Health and Human Services (HHS); the Gramm Leach Bliley Act, which pertains to financial institutions, is enforced by the Consumer Financial Protection Bureau, the FTC, and other financial regulators; Children's Online Privacy Protection, which protects children's data, is enforced by the FTC as well; and cyber standards for pipeline operators by the Transportation Security Administration.

The idea of federal government enforcement was less often discussed in the sampled literature than torts-based approaches, appearing in only 43 of the articles surveyed.

Visualizing articles' stances relative to their year of publication suggests that this idea emerged slightly later than did the idea of torts approaches and that it has gained relatievly more endorsements relative to mentions especially within the past decade. However, its only two criticisms have also occurred recently.

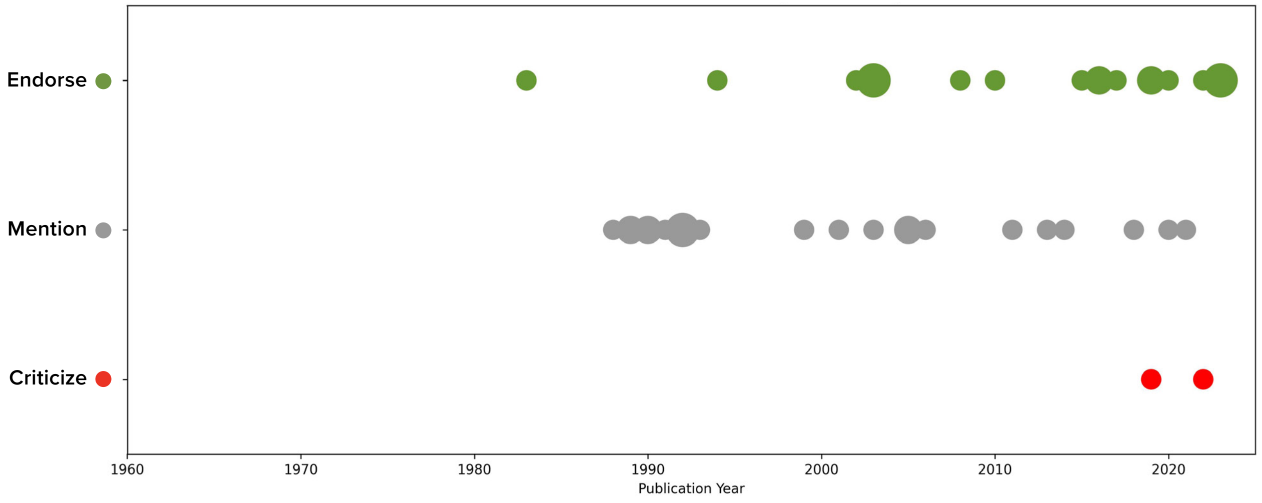
With respect to which agency or agencies should serve as an enforcer, in the data, only the FTC and the FDA (the latter typically within the context of medical devices) were named as potential federal enforcing entities in more than two articles. Additionally, 12 of the articles endorsed the idea of granting enforcement power to state law enforcement such as State Attorneys General.

Governance and Enforcement: A Federally Enforced Regime



Source: Cyber Statecraft Initiative

Governance and Enforcement: A Federally Enforced Regime



Consequences

Another key question is what happens to software makers that are found liable (and by whom). Most often, consequences come in the form of a requirement to pay money: either a penalty (for a violation of regulatory requirements) or damages (in torts, to compensate a harmed party). Tort-based regimes are necessarily civil, rather than criminal, proceedings; however, a statutory regime could create potential criminal liability with potential consequences including imprisonment. For example, violations of HIPAA, which regulates security controls for health care, can lead to both civil and criminal penalties, with criminal cases enforced by the Department of Justice rather than HHS.[25]

Regulatory regimes could draw from a few existing models to establish the monetary penalties to be applied for violations. They could structure the law as a penalty-per-violation—for example, the FTC can extract monetary penalties from entities that violate Children's Online Privacy Protection of up to $50,120 per violation.[26] In past, the FTC has extracted penalties in the hundreds of millions of dollars from the largest wrongdoers.[27] However, such regime would need to either set this per-violation cost to be very high or ensure that the number of violations is proportionate to the impact of the incident (for example, counting each separate instance of insecure software sold) in order to ensure that companies cannot walk away from a security failure that caused widespread harm with only a small fee to pay. Alternately, other regimes permit regulators to extract penalties based on the revenues of the penalized entity—for example, Europe's Cyber Resilience Act permits enforcers to extract penalties of up to 15 million euros or 2.5 percent of a company's total sales for the previous year, whichever is greater.[28] The European General Data Protection Regulation follows a similar model based on a percentage of the firm's worldwide annual revenue, with different tiers of possible fines depending on the specific provision violated.[29]

Under a regime structured using torts, companies would need to pay damages assessed by a judge. These damages can be "compensatory," or designed to compensate the impacted party for the harms they suffered, or "punitive," which damages are intended explicitly as a punishment above and beyond the harm caused. If implemented through torts, judges could draw upon a robust body of existing jurisprudence to determine appropriate compensation for harms arising from software insecurity; if instead achieved through a federal regime with a private right of action, lawmakers could tweak this penalty as well.

[25] "HIPAA Violations & Enforcement," American Medical Association, November 28, 2023, https://www.ama-assn.org/practice-management/hipaa/hipaa-violations-enforcement.

[26] "Complying with COPPA: Frequently Asked Questions," Federal Trade Commission, July 20, 2020, https://www.ftc.gov/business-guidance/resources/complying-coppa-frequently-asked-questions.

[27] "Google and YouTube Will Pay Record $170 Million for Alleged Violations of Children's Privacy Law," Federal Trade Commission, September 4, 2019, https://www.ftc.gov/news-events/news/press-releases/2019/09/google-youtube-will-pay-record-170-million-alleged-violations-childrens-privacy-law.

[28] "EU Cyber Resilience Regulation Could Translate into Millions in Fines." *Help Net Security (blog)*, January 19, 2023, https://www.helpnetsecurity.com/2023/01/19/eu-cyber-resilience-regulation-fines/.

[29] "What Are the GDPR Fines?," GDPR.eu, July 11, 2018, https://gdpr.eu/fines/.

Goals: What Does the Regime Try to Achieve?

Goals in the Literature

Each of the elements outlined above can be mixed and matched according to the goal of the liability regime. A liability regime must have a goal, explicit or implicit—or else, why effect a change? The rubric coded articles with respect to two broad-bucket potential goals. The first is to incentivize better security behavior by software vendors, typically in service of improving cybersecurity outcomes more broadly. The second is the question of providing redress, or ensuring that entities that are financially or otherwise harmed by a software vendors' failures are justly compensated. While the rubric coded only for these two goals, there are other possible ones, such as the desire to harmonize, unify, or preempt potentially diverse sets of cybersecurity requirements and liabilities that may emerge in the future under the evolution of common law doctrines or state law.

While these goals are not at all incompatible, they are also distinct—the fulfillment of one does not imply the fulfillment of the others. A regime could drive better software security without necessarily providing recompense to victims of insecurity, and vice versa. This section discusses each of the goals as represented in the literature and then explains how each goal might parameterize key design questions outlined above.

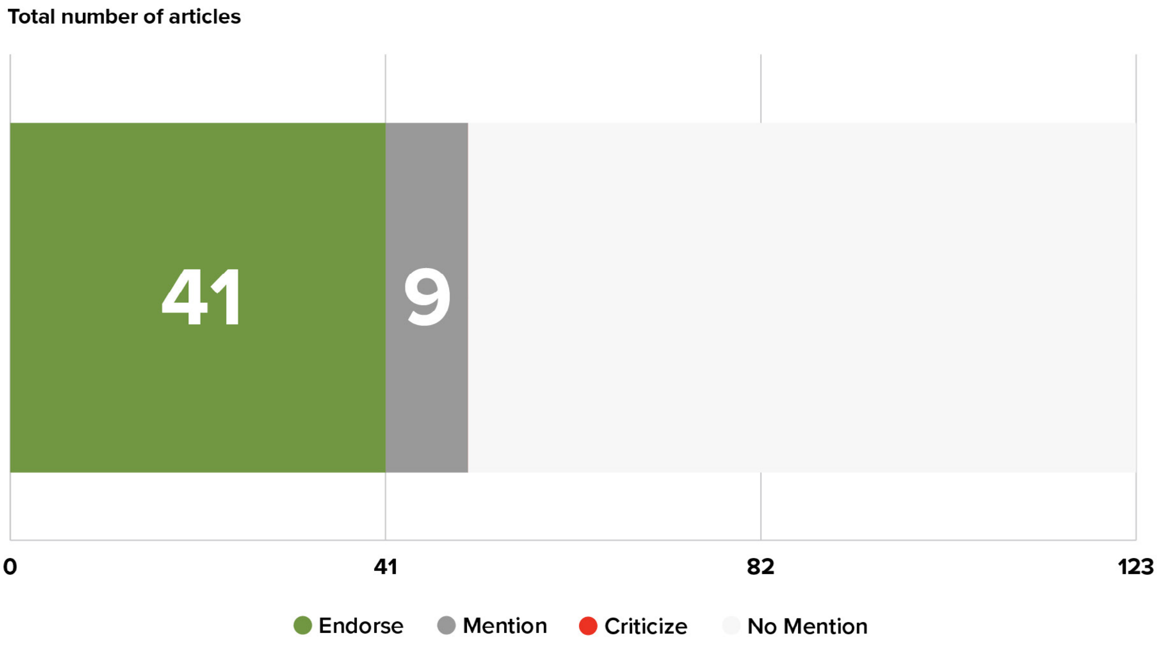
REBALANCE RESPONSIBILITY – INCENTIVIZE BETTER SECURITY

41 of the 123 surveyed articles described a potential goal for a liability regime in terms of changing incentives for software makers to to push them to adopt better security behaviors and practices.

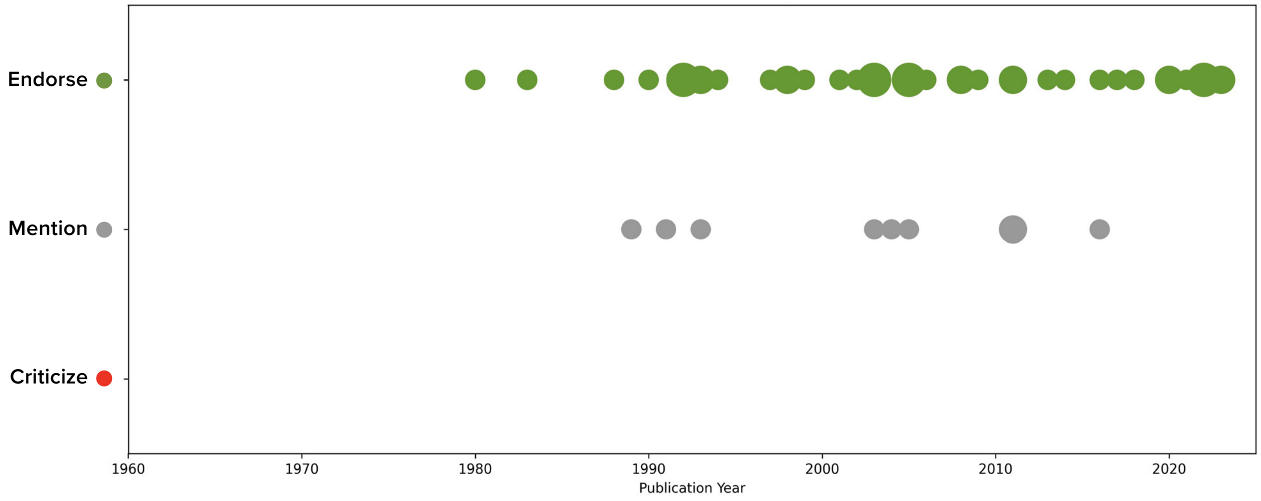
This goal has appeared in the sampled literature across multiple decades.

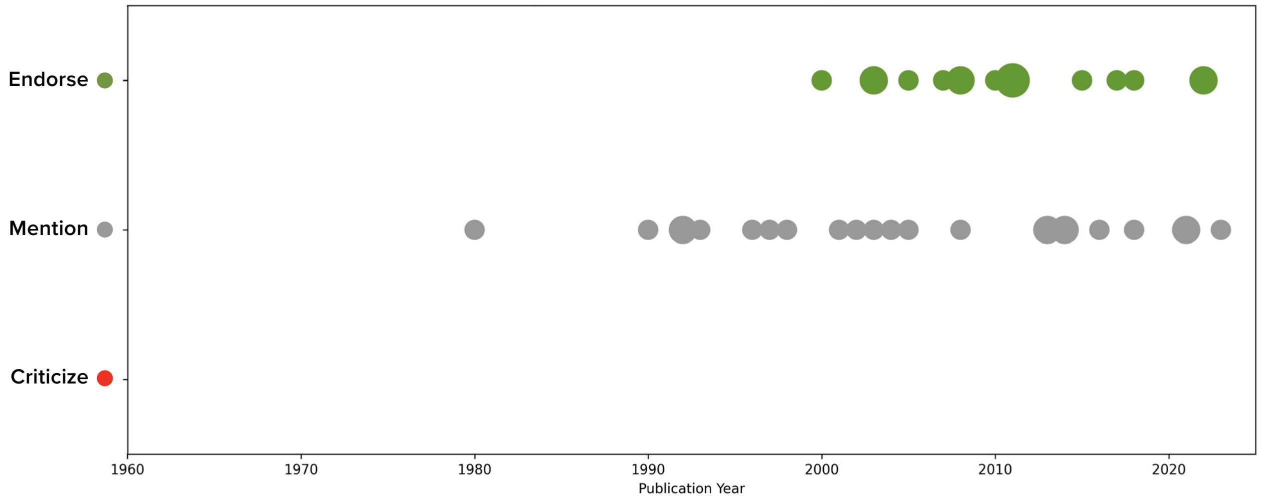
We expected this goal to be closely related to discussions of market failures or information asymmetries that limit the ability of the market to effectively incentivize better software security (e.g., the idea that software consumers are ill-positioned to evaluate the security of the software they buy

Goal: To Incentivize Better Security Behavior



Source: Cyber Statecraft Initiative


Goal: To Incentivize Better Security Behavior


Challenge: Market Failures or Information Asymmetries

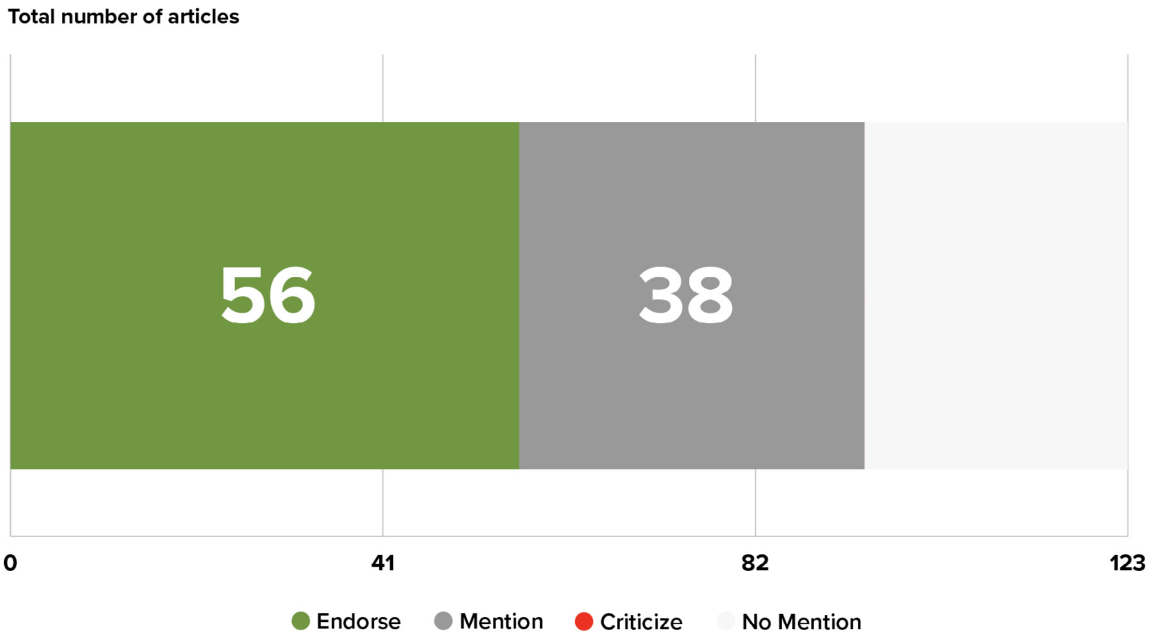
and thus the market inadequately incentivizes investment in security). Indeed, of the 41 articles with a stated goal of driving better security, ten explicitly cited market failures or information asymmetries as a current challenge with the ecosystem—a much higher rate than the six articles that endorsed this idea from the 130 remaining articles without such a goal. However, the idea of market failures and information asymmetries in security entered into the discussion in the surveyed literature relatively later, only after 2000.

PROVIDING REDRESS FOR HARMS

56 of the 123 articles explicitly endorsed the goal of providing redress for harmed software users as an explicit goal of a liability regime, with another 38 mentioning the idea without explicitly stating that it was a core goal or motivator for imposing a liability regime. That means this goal was present in more of the surveyed literature than that of improving security behavior and outcomes (though also more often mentioned without explicit endorsement).

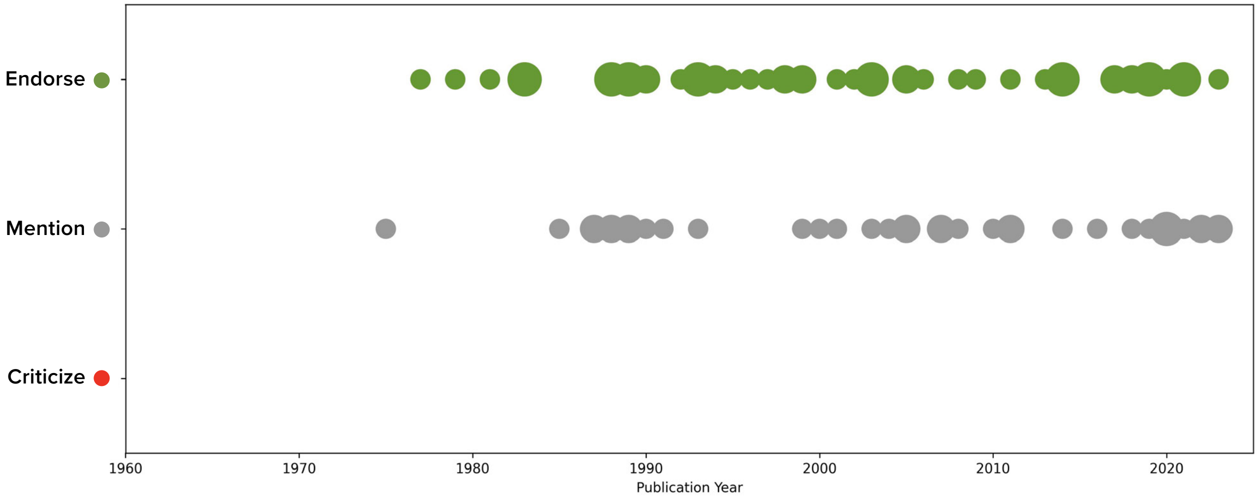
This goal also appears earlier than the goal of incentivizing better behavior in the sampled literature, first appearing as early as 1977.

Goal: To Provide Redress for Harmed Software Users



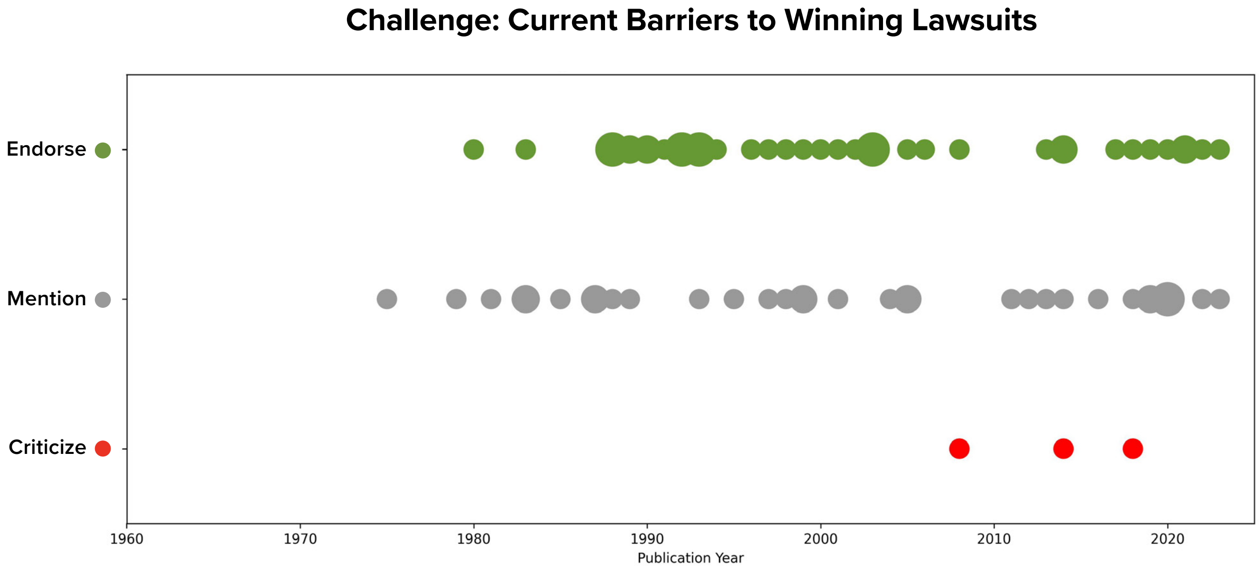
Source: Cyber Statecraft Initiative

Goal: To Provide Redress for Harmed Software Users



The goal of providing redress might reasonably be closely linked to the fact that currently, consumers and businesses struggle to recover losses from makers of insecure software. Of the 56 articles that endorsed providing redress to harmed parties as a core goal, 35 also mentioned current challenges and barriers to winning software cases, as compared to eight of the 115 articles that did not endorse providing redress as an explicit goal. Mention of difficulties winning current lawsuits have also been present in the corpus for nearly four decades:

This idea was emphasized during the 1990s, which may have been precipitated by ProCD, Inc. v. Zeidenberg, a court case that found so-called "shrink-wrap licenses"—licenses that the user "accepted" by opening the shrinkwrap that protected physical media like CDs that contained



software for install—to be legally valid.[30] However, this idea has continued to be mentioned throughout the articles in the years since.

Finally, the two goals are hardly incompatible: 25 articles explicitly endorsed both incentivizing security and redressing harm as a goal or motivation for imposing a liability regime.

Matching Goals to Other Elements of a Liability Regime

Different design choices in the construction of a liability regime will make the regime apply to different entities, incentivize different behavior, and provide different remedies. All these choices will shape its results and the changes it effects. Therefore, the explicit goal or goals of a liability regime provide direction on many of the key design choices outlined above.

REBALANCE RESPONSIBILITY – INCENTIVIZE BETTER SECURITY

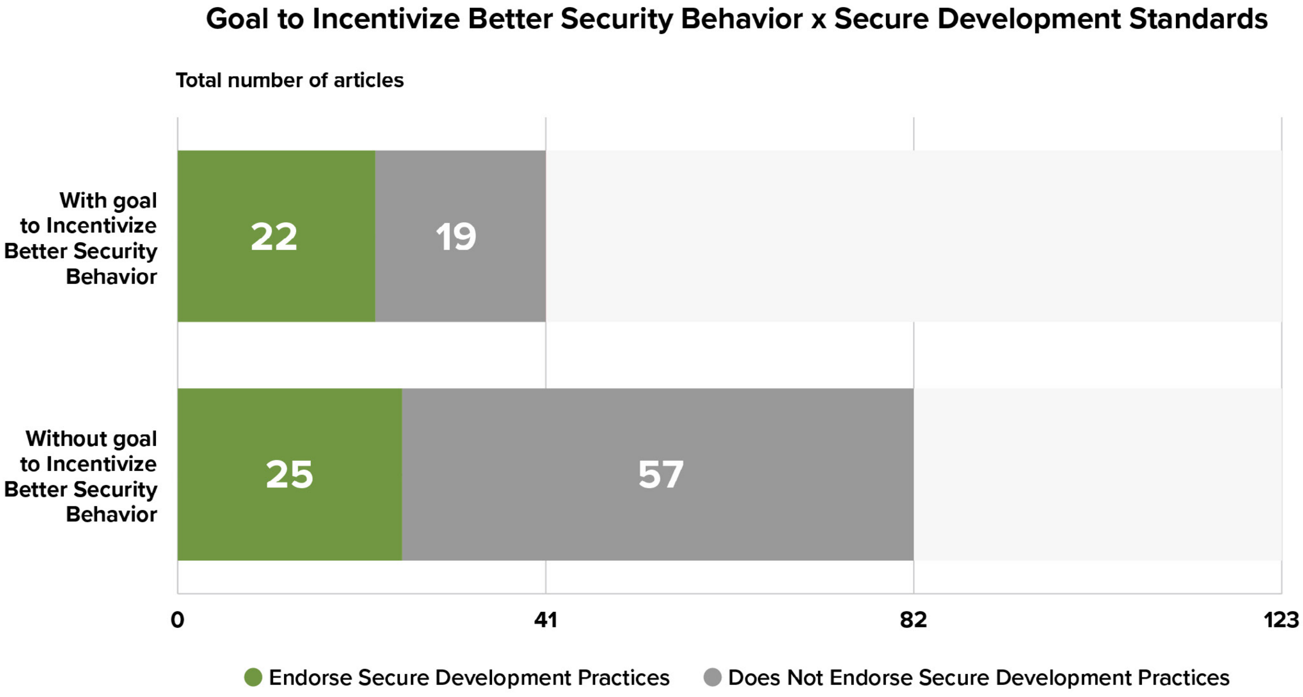
A regime designed to provide incentives for vendors to adopt more secure behavior is likely to focus strongly on the *standards* component of a regime, whether these standards are required in regulation or provide a safe harbor from tort liability. The standards baked into a regime will define the set of behaviors toward which software vendors will be incentivized, making it essential for policymakers with this goal to devise either strong standards, or a means for developing adaptive strong standards, which they believe will drive better security outcomes if adhered to.

Indeed, among articles from the literature in which the author identified incentivizing better security behavior as a core goal of a liability regime, a majority endorse the idea of including secure development standards as a component—not true for articles without such a goal. This substantiates the idea that there is a connection between the goal of improving security and a focus on the specific standards and practices that would need to be required in law to do so.

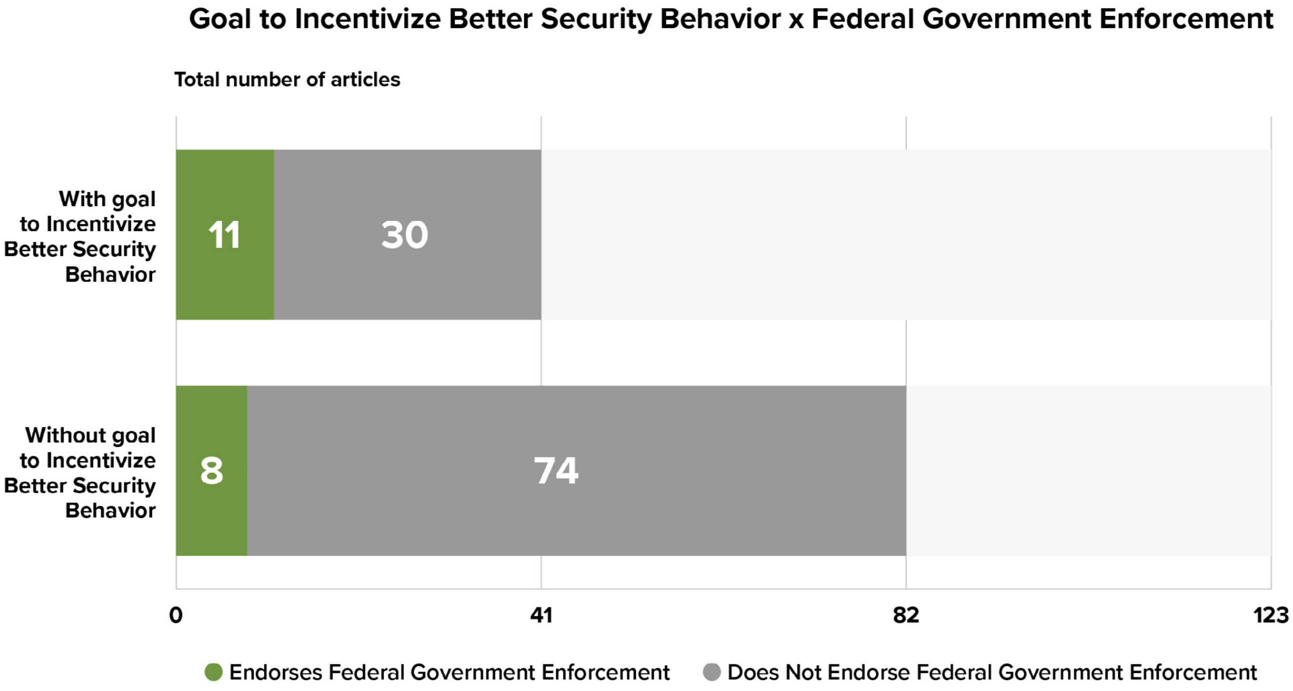
A goal of driving better security behavior might also make policymakers more interested in enforcement structures such as federal enforcement or torts liability with a safe or unsafe harbor, since these structures make it easier and faster to delineate clear standards through policy rather than waiting for courts to decide them over time. A regulatory regime might be particularly attractive for this goal because, unlike torts, it would not require harm to occur before action could be taken, potentially allowing enforcers to intervene before security malfeasance turns into individual or societal harm.

Indeed, a much larger percentage of articles with a goal to incentivize better security endorse federal enforcement, in contrast to articles that did not state an explicit goal of incentivizing better cybersecurity behavior.

[30] "ProCD, Inc. v. Zeidenberg," Casebriefs, accessed December 4, 2023, https://www.casebriefs.com/blog/law/contracts/contracts-keyed-to-farnsworth/the-bargaining-process/procd-inc-v-zeidenberg/.



Source: Cyber Statecraft Initiative



Source: Cyber Statecraft Initiative

Likewise, state government enforcement was more popular among articles that explicitly stated a goal of driving better security behavior. Delegating authority in such a way might increase the resources and enforcement power of the federal government, an appealing proposal for driving wider compliance. Surprisingly, governance mechanisms such as external auditing were less popular in articles with this goal than in the overall set, contravening the expectation that such measures would be popular because they would increase compliance and avoid the need to wait for a security incident to identify violators.

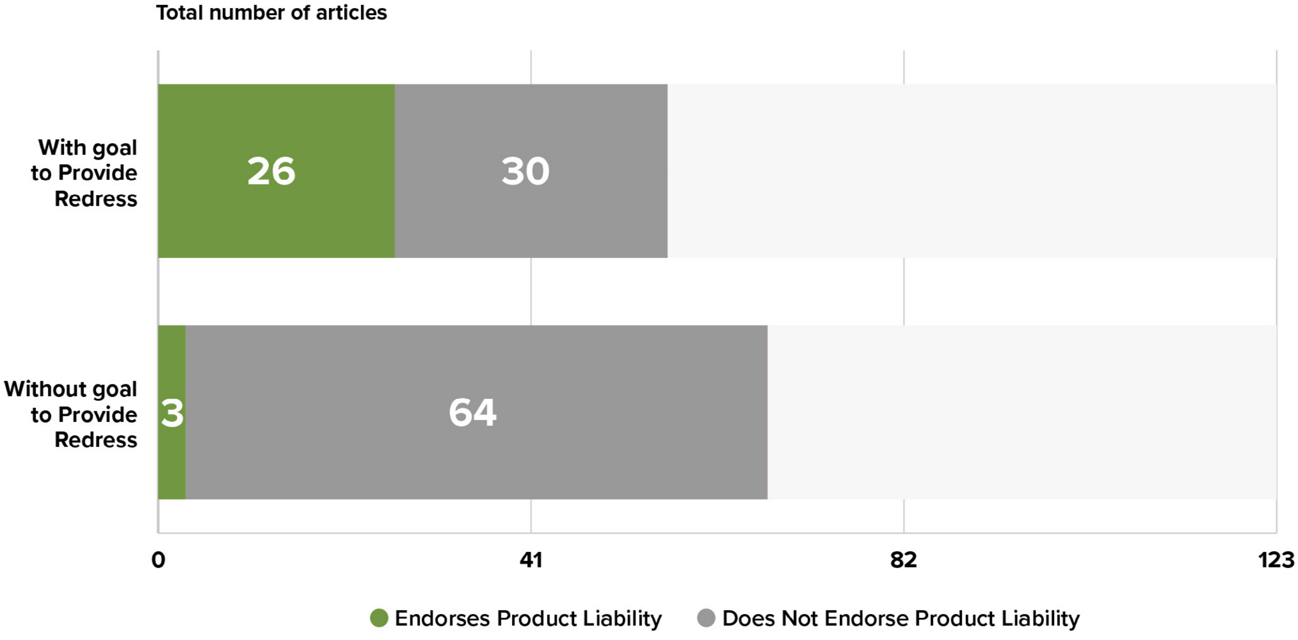
Articles with the goal of incentivizing better security behavior were more likely than those without to explicitly endorse either product or negligence liability regimes—for example, 16 out of 41 articles with this goal endorsed product liability as opposed to 13 out of 130 without the goal. Articles with this goal endorsed both product and negligence liability with approximately equal rates to the base set of literature.

PROVIDING REDRESS FOR HARMS

If the goal of a liability regime is to *provide redress* to users of software who were harmed by its insecurity, such a regime will be focused on the harms that can trigger liability, perhaps more so than on the specific standards that software makers must uphold. In fact, policymakers with this goal in mind might select a regime with very strict standards or even no standards at all to avoid cases in which harmed software users are denied redress because the software vendor met the legal baseline of responsible behavior.

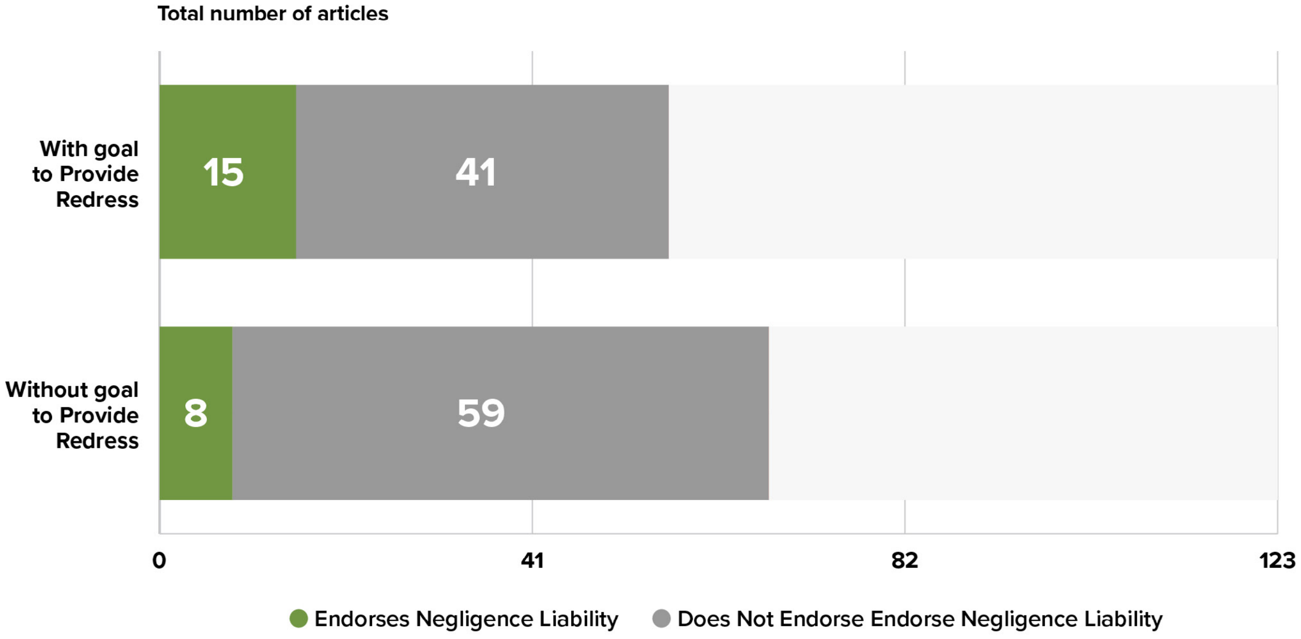
Indeed, sampled articles with a stated goal of providing redress for harmed users were much more likely to endorse strict product liability—more focused on whether the product itself was defective than on the manufacturer's intent—than articles without such a goal. These articles were also more likely to endorse strict product liability than negligence liability, which would incorporate a standard of care that defines software makers' obligations.

Goal to Provide Redress x Product Liability



Source: Cyber Statecraft Initiative

Goal to Provide Redress x Negligence Liability



Source: Cyber Statecraft Initiative

Limitations and Directions for Future Work

The sample of the literature conducted herein has several limitations that could be improved on in future work. First, the selection of articles was based on keyword searches and expert judgement rather than a measure such as citation count for all articles, which limits our ability to understand whether the sample is representative of the broader debate. Second, several factors of particular interest in this debate resolved only after the coding was completed, meaning the rubric did not incorporate some relevant questions such as limiting the applicability of a regime by type of software product or how to handle different types of supply chain compromise. Future work might consider a more robust methodology for article selection and a more extensive rubric. It might also lessen the degree of subjectivity in the actual coding by codifying standards and examples of endorsement, mention, and criticism ahead of time, or by having multiple reviewers code the same article and then using the average of their judgements to inform a final score.

Conclusion

Conducting a meta-analysis of a complicated debate such as software liability necessarily produces data that is more illustrative than it is dispositive. The trends outlined above are not meant to present definitive answers as to the right approach on liability, but instead to provide a structuring framework that can help illuminate how different policy design questions—and the relationships between such questions—have been discussed (and sometimes under-discussed) thus far in the scholarly debate. In particular, some of the topics that were relatively more neglected in the literature sample, such as specific frameworks that could form the basis for standards in a liability regime, how to handle the problem of user behavior and patching, and how to scope the regime or its standards to different sectors or types of software, seem to be areas where further study and debate are much needed.

Though it is tempting to analogize software liability neatly to other products or goods for which policymakers have constructed successful liability regimes—a popular metaphor is cars—these metaphors obscure important details of the ways that software is meaningfully different and poses greater challenges to regulate as a class of technology than many that have come before. Software is everywhere—it is found in every industry, in every application from the most trivial to the most consequential. It ranges almost unimaginably in scale and complexity, from tiny calculator applications to vast and sprawling networks of cloud computing infrastructure on which an incalculable number of other computing applications depend. These factors create paradoxes for regulators: societally and economically, we benefit hugely from a fast-moving, innovative, and thriving ecosystem of software development. At the same time, the persistent ethos of "ship now, fix later" has led to vulnerabilities that have cost collective billions of dollars[31] and damaged individual privacy[32] and national security[33] through myriad cyber incidents great and small.

In practice, software liability may not be realized through a single, comprehensive regime that addresses every concern and every type of software at once. Instead, it might be an incremental form of progress: the creation of a duty of care for the largest vendors, or a requirement for the majority to adopt a small set of known best practices. One key throughline is likely to be adaptability: the ability of a regime to adapt to evolving best practices in the software security landscape, to adapt standards to different paradigms and functions for software, and to adapt to the different scales and stakes of various software applications.

The policy task ahead on software liability is complex and contested—it will demand common language in addition to common purpose. This work brings forward a set of core design questions from the history of the debate to help advance the current policy conversation around software liability, all in service of one goal: to improve outcomes in a world ever more reliant on the security of the software it consumes.

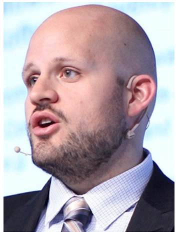
[31] Annie Lowrey, "Sony's Very, Very Expensive Hack." *New York Magazine*, December 16, 2014, https://nymag.com/intelligencer/2014/12/sonys-very-very-expensive-hack.html.

[32] Federal Trade Commission, "Equifax to Pay $575 Million as Part of Settlement with FTC, CFPB, and States Related to 2017 Data Breach," July 22, 2019, https://www.ftc.gov/news-events/news/press-releases/2019/07/equifax-pay-575-million-part-settlement-ftc-cfpb-states-related-2017-data-breach.

[33] Trey Herr et al., "Broken Trust: Lessons from Sunburst," *Atlantic Council (blog)*, March 29, 2021, https://www.atlanticcouncil.org/in-depth-research-reports/report/broken-trust-lessons-from-sunburst/.

About the Authors

Maia Hamin is an Associate Director with the Atlantic Council's Cyber Statecraft Initiative under the Digital Forensic Research Lab (DFRLab). She works on the intersection of cybersecurity and technology policy, including projects on the cybersecurity implications of artificial intelligence, open-source software, and cloud computing. Prior to joining the Council, Maia was a TechCongress Congressional Innovation Fellow serving in the office of Senator Ron Wyden, and before that a software engineer on Palantir's Privacy and Civil Liberties team. She holds a B.A. in Computer Science from Princeton University.



Dr. Trey Herr is the director of the Atlantic Council's Cyber Statecraft Initiative and an assistant professor of Cybersecurity and Policy at American University's School of International Service. At the Council, the CSI team works at the intersection of cybersecurity and geopolitics across conflict, cloud computing, supply chain policy, and more. Previously, he was a senior security strategist with Microsoft handling cloud computing and supply chain security policy as well as a fellow with the Belfer Cybersecurity Project at Harvard Kennedy School and a non-resident fellow with the Hoover Institution at Stanford University. He holds a PhD in Political Science and BS in Musical Theatre and Political Science.

Sara Ann Brackett is a research associate at the Atlantic Council's Cyber Statecraft Initiative under the Digital Forensic Research Lab (DFRLab). She focuses on open-source software security (OSS), software bills of materials (SBOMs), software liability, and software supply-chain risk management within the Initiative's Systems Security portfolio. She is an undergraduate at Duke University, where she majors in Computer Science and Public Policy, participates in the Duke Tech Policy Lab's Platform Accountability Project, and works with the Duke Cybersecurity Leadership Program as part of Professor David Hoffman's research team.

Andy Kotz is a recent graduate of Duke University, where he majored in Computer Science and Political Science and served as a Cyber Policy Research Assistant on Professor David Hoffman's research team.

[redacted]

The Atlantic Council is a nonpartisan organization that promotes constructive US leadership and engagement in international affairs based on the central role of the Atlantic community in meeting today's global challenges.

1030 15th Street, NW, 12th Floor,
Washington, DC 20005
(202) 778-4952
www.AtlanticCouncil.org