LAWFARE
HARD NATIONAL SECURITY CHOICES

SECURITY BY DESIGN

Investing in Rust

Shane Miller*

JULY 2024

Research consistently attributes more than 50 percent of security vulnerabilities to errors that are prevented by using memory-safe programming languages. Despite those benefits, adoption of memory-safe languages is stalled in some domains, because memory-unsafe languages like C and C++ have locked in the market. Unlike older memory-safe languages such as Java or Python, the relatively new Rust language optimizes efficiency with memory safety. Unfortunately, Rust's innovative design and implementation are incompatible with existing engineering skills and systems, creating market friction for adoption. This paper recommends U.S. public policy to mitigate that friction and foster the adoption of memory-safe languages.

INTRODUCTION

In February 2024, a cyberattack on UnitedHealthcare Group threatened the solvency of thousands of U.S. hospitals, sent "a substantial proportion" of Americans' medical records into the dark web,[1] and prevented untold thousands of patients from receiving their prescriptions. Some 94 percent of U.S. hospitals were financially impacted, with nearly 60 percent reporting daily losses over a million dollars.[2] One leader of an Idaho medical center devastated by this attack called it "a bigger deal financially than Covid."[3]

This breach is just the latest in what's become routine: cyberattacks taking advantage of the increasing fragility of America's critical technology. At least 299 U.S. hospitals reported

***Shane Miller** is a Distinguished Advisor to the Rust Foundation, where she was the founding chair of the board of directors. Miller is also a senior fellow at the Atlantic Council Cyber Statecraft Initiative under the Digital Forensic Research Lab and an advisory board member for the State of Open Con. She is the former founding leader of four different organizations at Amazon Web Services (AWS), including Rust open source.

[1] Zack Whittaker, "UnitedHealth Says Change Hackers Stole Health Data on 'Substantial Proportion of People in America,'" *TechCrunch*, April 22, 2024, https://techcrunch.com/2024/04/22/unitedhealth-change-healthcare-hackers-substantial-proportion-americans/.

[2] Noah Barsky, "UnitedHealth Paid Hackers $22 Million, Fixes Will Soon Cost Billions," *Forbes*, June 7, 2024, https://www.forbes.com/sites/noahbarsky/2024/04/30/unitedhealths-16-billion-tally-grossly-understates-cyberattack-cost/.

[3] John Sakellariadis, "Hospitals Are Pleading for Help. The NSC May Be Close to Giving It," *Politico*, March 4, 2024, https://www.politico.com/newsletters/weekly-cybersecurity/2024/03/04/hospitals-are-pleading-for-help-the-nsc-may-be-close-to-giving-it-00144647.

cyberattacks last year alone,[4] and health care is not a uniquely vulnerable sector. Some 80 percent of American school administrators say they have been the victims of ransomware attacks,[5] causing schools to use "snow day" budgets for "cyber day" closures.[6] Major companies—Walmart, Samsung, 23andMe, Microsoft, MGM Grand, Discord, T-Mobile, ChatGPT—have reported catastrophic breaches. And that's just in the past year.

In response to these growing attacks, the White House has established an agenda to improve the cybersecurity of critical American infrastructure, launching a series of executive orders,[7] strategies,[8] implementation plans,[9] and directives.[10] One of the consistent cornerstones of the White House cybersecurity campaign is addressing memory-safety classes of vulnerabilities, saying, "We, as a nation, have the ability—and the responsibility—to reduce the attack surface in cyberspace and prevent entire classes of security bugs from entering the digital ecosystem but that means we need to tackle the hard problem of moving to memory safe programming languages."[11]

Memory-safe programming languages prevent software engineers from making errors that are frequently exploited by malicious actors, and that prevention has an outsized impact on software security. Several industry analyses have concluded that memory-safe languages avoid more than half of all security vulnerabilities, with both Microsoft[12] and Google[13] research attributing 70 percent of security vulnerabilities to using memory-unsafe languages. In addition to security benefits, memory-safe languages reduce software maintenance expenses and improve engineering agility. At the same time, factors that influence adoption of new technologies are slowing the spread of memory-safe languages in some domains. Public-private

[4] Nicole Sganga, "Latest Hospital Cyberattack Shows How Health Care Systems' Vulnerability Can Put Patients at Risk," CBS News, Nov. 29, 2023, https://www.cbsnews.com/news/ardent-hospital-cyberattack-health-care-system-vulnerability/.
[5] Lauraine Langreo, "7 Data Breaches That Left Schools in the Lurch," *Education Week*, Aug. 17, 2023, https://www.edweek.org/technology/7-data-breaches-that-left-schools-in-the-lurch/2023/08.
[6] Kavitha Cardoza, "One Reason School Cyberattacks Are on the Rise? Schools Are Easy Targets for Hackers," National Public Radio *All Things Considered*, March 11, 2024, https://www.npr.org/2024/03/11/1236995412/cybersecurity-hackers-schools-ransomware.
[7] Executive Office of the President [Joseph Biden]. Executive Order 14028: Improving the Nation's Cybersecurity, May 12, 2021. Federal Register, vol. 86, no. 2021-10460, pp. 26633–47, https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity.
[8] The White House, *National Cybersecurity Strategy*, March 1, 2023, https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf.
[9] The White House, *National Cybersecurity Strategy Implementation Plan*, July 13, 2024, https://www.whitehouse.gov/wp-content/uploads/2023/07/National-Cybersecurity-Strategy-Implementation-Plan-WH.gov_.pdf.
[10] The White House, Office of the National Cyber Director, "Future Software Should Be Memory Safe," Press Release, Feb. 26, 2024, https://www.whitehouse.gov/oncd/briefing-room/2024/02/26/press-release-technical-report/.
[11] Ibid.
[12] Sebastian Fernandez, "A Proactive Approach to More Secure Code," Microsoft, July 16, 2019, https://msrc.microsoft.com/blog/2019/07/a-proactive-approach-to-more-secure-code/.
[13] Chromium Security, "Memory Safety," Google, https://www.chromium.org/Home/chromium-security/memory-safety/.

partnerships can address market friction with initiatives that highlight the business benefits and lower the cost, complexity, and risk of memory-safe languages. Memory safety is good for businesses and consumers, and strategic policies and investments can make it better.

Despite strong guidance in the past few years from agencies such as the National Security Agency, which "recommend[ed] that organizations use memory safe languages when possible,"[14] little has changed. Analyst firm Redmonk's 2023 language report noted, "The dominant trend [is still] lack of movement. While the industry around these programming languages is evolving rapidly, the inertia of language traction has proven difficult to overcome."[15] **Memory-unsafe programming languages are not losing ground.** The TIOBE Index,[16] which measures programming language popularity, found that increases in C++ were equivalent to decreases in C from 2020 to 2024, keeping the overall popularity of memory-unsafe programming languages unchanged, and other reputable indexes like PYPL[17] report similar trends.

Unsafe code remains prolific because (a) memory safety was added to programming language design long after engineers started building the software foundational to modern technology, giving memory-unsafe languages a huge head start; (b) until Rust became viable, software could not use memory-safe languages everywhere; and (c) market friction is slowing the adoption of Rust. Historically, safe languages (like Java, Python, and JavaScript) have produced slow systems that consume far more resources than their unsafe predecessors. For resource-restricted solutions like mobile devices and the networks that connect them, safe languages have not been an option. The relatively new Rust language offers a solution that combines the optimized efficiency of memory-unsafe languages like C and C++ with the security of modern memory safety. As a result, there are far fewer cases where using a memory-safe language is not possible today, because technology manufacturers no longer need to sacrifice security for efficiency.

Rust is a new memory-safe language that can be used for many solutions previously without a memory-safe option, like cloud computing and operating systems that require optimized performance and resources. Rust achieves this by enforcing memory safety at compile time, whereas other memory-safe languages (like Java, Python, and JavaScript) use a feature called a "garbage collector" to manage memory while software is running. The garbage collector handles the challenges and risks of memory management for the engineer, but it requires far more resources[18] and forces systems to pause periodically for cleanup. Rust delivers memory safety without resource and performance penalties. To make that possible, Rust developers

[14] U.S. National Security Agency, "Software Memory Safety," Nov. 10, 2022, https://media.defense.gov/2022/Nov/10/2003112742/-1/-1/0/CSI_SOFTWARE_MEMORY_SAFETY.PDF.
[15] Stephen O'Grady, "The RedMonk Programming Language Rankings: January 2023," Redmonk, May 16, 2023, https://redmonk.com/sogrady/2023/05/16/language-rankings-1-23.
[16] TIOBE Index, TIOBE, https://www.tiobe.com/tiobe-index/.
[17] PYPL Index, PYPL PopularitY of Programming Language, https://pypl.github.io/PYPL.html.
[18] Matthew Hertz and Emery D. Berger, "Quantifying the Performance of Garbage Collection vs. Explicit Memory Management," *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '05*, pp. 313–26. doi: 10.1145/1094811.1094836.

must follow strict coding rules that ensure memory is managed correctly in their software, and the Rust compiler that transforms code into executable software rejects code that does not adhere to those rules.

While Rust's innovation fills a critical memory-safety gap, the language's design and implementation are incompatible with existing engineering skills and systems, creating substantial market friction for technology manufacturer adoption. **Rust adoption is slowed by four of the five factors influential in the diffusion of a new idea or innovation.**[19] In addition to (a) incompatibility and (b) its accompanying complexity, Rust struggles with (c) the observability of adoption and (d) relative advantage. Relative advantage is the perceived improvement of an innovation, and adoption is faster when relative advantage is high. Preventive innovations like memory safety that lower the likelihood of a negative future event have a particularly slow rate of adoption, because the reward is far in the future without clear evidence of causality.[20]

Like any new technology, software made with Rust is also more expensive to build. Rust developer salaries are among the highest paid,[21] training existing engineers is challenging (only 47 percent of surveyed Rust engineers consider themselves productive using the language), and building support infrastructure for the thousands of open source projects used to write Rust is a substantial investment. At the same time, there is strong evidence that Rust lowers the cost of software ownership by reducing maintenance costs over its total lifetime. Building software can take months or even years, but if that software is successful and customers use it, maintaining the software will take decades. As Amazon Web Services Distinguished Engineer Marc Brooker said in 2020, "Software lasts a long time …. Initial development is the easy, relatively cheap part of building a system, and the expensive part … is maintaining it in production."[22]

While software costs may be amortized by customers over just a few years, that software is often used in production for significantly longer, because replacing business-critical software is expensive and risky. As software becomes larger and more complex, the costs and risks of replacing it grow. Researchers studying organizational behavior and sentiment with legacy systems noted the common opinion that "by definition a legacy system is business critical. A system that is old and obsolete and is not business critical would never reach the status of legacy."[23] Frequently, only parts of legacy systems are replaced over time, while the original software persists in production for some functionality.

[19] Everett M. Rogers, *Diffusion of Innovations*, 5th ed. (Free Press, 2003), 266.
[20] Ibid, 234.
[21] Afifa Mushtaque, "5 Highest-Paying Programming Languages in USA," Insider Monkey, July 17, 2023, https://www.insidermonkey.com/blog/5-highest-paying-programming-languages-in-usa-1168666/4/.
[22] Marc Brooker, "Building Technology Standards at Amazon Scale," YouTube, uploaded by AWS Events, Feb. 5, 2021, https://youtu.be/2xoNsusfOyE?si=zMKt528CF27Ev3-1.
[23] Ravi Khadka, Belfrit Batlajery, Amir Saeidi, et al., "How Do Professionals Perceive Legacy Systems and Software Modernization?" ACM *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 36–47. doi: 10.1145/2568225.2568318.

Unfortunately, the maturity of our legacy technology is not yielding better security. Detailed reviews of mature open source software like the Linux operating system distribution Debian, the programming language PHP, and the Java developer platform OpenJDK have found that security does not improve over time and more generally that “we have not reached the point of curbing the vulnerability rate.”[24] The new Rust programming language makes memory safety possible in far more technologies, preventing most security vulnerabilities from being created in the first place.

At the same time, Rust is open source, challenged with supply chain risks and complexities common to all community-supported software. Moreover, programming languages like Rust are a subset of open source, and other open source projects like the operating system Linux are implemented using those languages. This circular dependency requires analysis that considers both Rust-specific issues and more general issues relating to open source software that impact Rust. One of those challenges is the growing tension between China and the United States.

While U.S. policymakers have restricted China’s access to advanced hardware[25] and passed a law demanding that the Chinese company ByteDance sell TikTok or stop operating the mobile app in the United States,[26] almost all U.S. technology is built by teams composed of both China- and U.S.-funded engineers. Some 96 percent of technology includes open source, and the technologies that use open source are made up primarily of open source software. Detailed scans across a wide variety of critical industries, such as health care, finance, and transportation, found that 77 percent of the software’s code originates from open source, and open source is as Chinese as it is American.[27]

China protects its software supply chain from political intervention[28] and malicious interference (like the recent XZ Utils attack)[29] by providing a government-funded, quality-controlled copy of open source for Chinese corporations and developers.[30] Open source projects like Rust have

[24] Nikolaos Alexopoulos, Sheikh Mahbub Habib, Steffen Schulz, and Max Mühlhaüser, “The Tip of the Iceberg: On the Merits of Finding Security Bugs,” ACM *Transactions on Privacy and Security* 24, no. 1 (2021): 1–33. doi: 10.1145/3426975.
[25] Josh Boak, “The Commerce Department Updates Its Policies to Stop China From Getting Advanced Computer Chips,” Associated Press, Oct. 17, 2023, https://apnews.com/article/computer-chips-export-china-biden-raimondo-78225ba8d1609137e859f68a80f6e91e.
[26] Bobby Allyn, “President Biden Signs Law to Ban TikTok Nationwide Unless It Is Sold,” National Public Radio, April 24, 2004, https://www.npr.org/2024/04/24/1246663779/biden-ban-tiktok-us.
[27] Fred Bals, “2024 Open Source Security and Risk Analysis Report,” Synopsys, Feb. 27, 2024, https://www.synopsys.com/blogs/software-security/open source-trends-ossra-report.html.
[28] Rita Liao, “China Is Building a GitHub Alternative Called Gitee,” *TechCrunch*, Aug. 21, 2020, https://techcrunch.com/2020/08/21/china-is-building-its-github-alternative-gitee/.
[29] Kevin Roose, “Did One Guy Just Stop a Huge Cyberattack?” *New York Times*, April 3, 2024, https://www.nytimes.com/2024/04/03/technology/prevent-cyberattack-linux.html.
[30] Coco Feng, “Gitee, China’s Answer to GitHub, to Review All Code by Temporarily Closing Open source Projects to the Public,” *South China Morning Post*, May 19, 2022, https://www.scmp.com/tech/big-tech/article/3178323/gitee-chinas-answer-github-review-all-code-temporarily-closing-open.

become a global public good worth nearly $9 trillion,[31] and U.S. public policy can have historic economic security impact by addressing risks to important open source work like memory-safe languages that are foundational to the resilience of critical infrastructure as well as the market hurdles that stall adoption of emerging solutions like Rust.

What is needed now is a jump-start. This paper outlines a policy proposal that provides for

- an addition to the critical infrastructure information technology sector,
- a cloud computing tax to fund critical U.S. cyber defense,
- U.S.-sponsored governance for emerging cybersecurity solutions like Rust, and
- a U.S.-sponsored open source library verification service.

Secure-by-design must include memory-safe-by-default, and memory-safe-by-default needs secure and accessible memory-safe programming languages. Public policy must extend beyond driving adoption of memory-safe languages to supporting the security and stability of them. Before elaborating on these policy objectives in greater detail, this paper provides a more detailed explanation of memory-safe languages and, particularly, the significant improvements arising from the use of the Rust language.

RUST LOWERS THE TOTAL COST OF SOFTWARE OWNERSHIP

Memory-safe languages reduce the effort required for operations, freeing engineers to focus on building a new version or feature for their product. Modern software development is an iterative process in which product launch is not the end of the effort but the beginning of operations roles and responsibilities for the engineering team.

After a product launch, engineering resources cannot focus exclusively on building new features because attention and capacity must be split between building the next thing and operating the existing one. Operating costs have a huge impact on the team's ability to stay agile and competitive. Operational tasks like on-call rotations[32] and fixing suboptimal software[33] can adversely impact team morale and performance in all areas if they are not managed and contained. Maintaining and evolving software includes engineering work for

- monitoring and reacting to system operations,
- patching security vulnerabilities,
- fixing bugs, and
- adding new features to support evolving user needs.

[31] Rachel Layne, "Open Source Software: The $9 Trillion Resource Companies Take for Granted," Harvard Business School Working Knowledge, March 22, 2024, https://hbswk.hbs.edu/item/open source-software-the-nine-trillion-resource-companies-take-for-granted.
[32] Grace E. Vincent, Katya Kovac, Leigh Signal, et al., "What Factors Influence the Sleep of On-Call Workers?" *Behavioral Sleep Medicine* 19, no. 2 (2021): 255–72. doi: 10.1080/15402002.2020.1733575.
[33] Terese Besker, Hadi Ghanbari, Antonio Martini, and Jan Bosch, "The Influence of Technical Debt on Software Developer Morale," *Journal of Systems and Software* 167 (2020). doi: 10.1016/j.jss.2020.110586.

Rust's newness and immaturity increase the initial time and cost of software development, but its combination of efficiency and memory safety decrease the substantially larger cost of maintenance and operations that consumes 60 to 70 percent of an engineering organization's resources.[34] The quality and security improvements baked into software built with memory-safe programming languages have delivered 60 percent fewer memory-safety vulnerabilities,[35] 75 percent fewer bugs,[36] and ten times fewer failures[37] for early memory-safety migrations, empowering software engineers to iterate and refactor more, keeping software and its dependencies current throughout its lifetime. Using a conservative assumption of a 50 percent reduction in effort for operations, security patching, and bug fixing, memory safety frees 30 percent (half of 60 percent) of an engineering team's maintenance capacity. For a one-hundred-person engineering organization, that means memory-safe languages can nearly double the number of engineers working on new features and products, from forty to seventy people.

Monitoring and Reacting to System Operations

Before software is made available to customers, engineering teams create dashboards with metrics that provide visibility into the operations of their software and set alarms that go off in the event of failures. They create "on-call" rotations for first responders and runbooks with protocols those operators will follow. Software errors require immediate attention when they impact customers, and engineering operations assume systems will fail. New users and increased traffic interact with software in ways its authors did not predict, revealing latent bugs and security vulnerabilities. The operational controls the team sets up will identify some of those challenges, while manual reports like emails or customer service calls will catch others. The engineering team will review reports as it receives them, classify them through a triage process, and establish a plan to fix or mitigate them. In many cases, the team will change the software with a fix to the original code, and in some cases, the team will also update its operations tools and processes.

Amazon Prime Video observed huge reductions in the frequency of those operational errors with their memory-safe migration. Amazon's Prime Video team migrated from JavaScript to Rust and WebAssembly for performance improvements. As part of that migration, the team also replaced a portion of unsafe C++ code with Rust. Adopting memory-safe code significantly improved service reliability. **The crash rate for Amazon's new Rust software is ten times**

[34] Andrea Bordin and Fabiane Barreto Vavassori Benitti, "Software Maintenance: What Do We Teach and What Does the Industry Practice?" *XXXII Brazilian Symposium on Software Engineering*, 2018. doi: 10.1145/3266237.3266251.
[35] Jeffrey Vander Stoep, "Memory Safe Languages in Android 13," Google Security, Dec. 1, 2022, https://security.googleblog.com/2022/12/memory-safe-languages-in-android-13.html.
[36] Adam Zabrocki and Alex Tereshkin, "Exploitation in the Era of Formal Verification," YouTube, uploaded by DEFCONConference, Oct. 20, 2022, https://youtu.be/TcIaZ9LW1WE?si=21o29TQtPp9Uo75n.
[37] Alexandru Ene, "Optimizing Prime Video With WebAssembly and Rust," YouTube, uploaded by International JavaScript Conference, Sept. 20, 2022, https://youtu.be/erdHTxghyM0?si=2tRL_7u8EbjNW7KO.

smaller than for their C++ systems.[38] The Prime Video Rust and C++ services are authored and operated by the same team of engineers, the code bases are relatively similar in size, and they're following industry best practices for preventing, detecting, and correcting memory errors in their C++ code, providing a useful baseline for comparison. Reducing crashes by ten times reduces the number of alarms Amazon's systems sound, the number of times the engineering team must triage a crash, and the number of fixes the team must apply to its software. That substantially lowers the cost of operations, creates a better experience for users, and gives the team more capacity to build profitable new features.

Patching Security Vulnerabilities

Some of the most urgent and important software errors will be exploitable vulnerabilities. The operational savings that companies like Amazon reap from safe programming languages come as a wonderful side effect of the memory safety baked into those languages to ensure correctness. Memory-safe languages are the most effective and cost-efficient protection from malicious activity because most security vulnerabilities are errors that are not possible with memory-safe programming languages. Despite a surge in security and developer tools, education, and process investments over the past couple of decades, open source software security research suggests that "we have not yet achieved an adequate degree of rigorousness in our development and security processes … [because] the number of [security] vulnerabilities [in software] does not visibly decrease over time, even for software that has been stable for many years."[39]

Researchers find that software vulnerabilities identified in new releases are not overwhelmingly new but often residual bugs present in previous releases and identified only as the result of fresh security examinations triggered by a new release. Each vulnerability goes through an operations engineering team process. It is reported and triaged, before a fix to some code is implemented, tested, and deployed. Some users update their software and receive that fix, while others continue to operate exploitable versions. **Implementing software with memory-safe languages prevents most security vulnerabilities from ever being created.** Instead of spending valuable engineering time finding and fixing these security vulnerabilities in production, where they are most expensive, memory-safe software is secure by design.

The Google Android team has been taking advantage of those benefits, because their team found that memory errors by developers in C and C++ code disproportionately accounted for their most dangerous security vulnerabilities. In 2022, lack of memory safety accounted for 86 percent of critical severity vulnerabilities and 89 percent of remotely exploitable vulnerabilities. Over the past several years, 78 percent of confirmed exploited vulnerabilities on Android devices were memory bugs.[40]

[38] Ibid.

[39] Alexopoulos et al., supra note 24.

[40] Vander Stoep, supra note 35.

The Google Android team is transitioning to memory-safe programming languages like Java, Kotlin, and Rust, and more than half of the new code in Android version 13 was written with those safe languages. The result of the transition has been consistent drops in memory-safety vulnerabilities as well as the severity of vulnerabilities still reported, with vulnerabilities reduced more than 60 percent over the past four years (see Figure 1). That's 60 percent fewer security fire drills for the Google Android team, because they started using memory-safe programming languages by default. That saves the Google Android team substantial money and time that they can invest in building new features (like upgraded camera and media options) for their product.

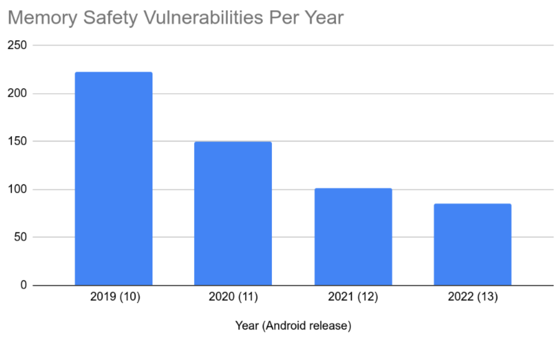


Figure 1. Google Android memory-safety vulnerabilities.

Fixing Bugs

The Rust compiler prevents software engineers from unknowingly producing code with memory-safety bugs, and that improves both the security and the resilience of the software Rust is used to build. Fixing bugs earlier in the development life cycle is also substantially cheaper because resources and processes accumulate as software moves through the development life cycle, slowing down changes and exponentially increasing their cost. Figure 2 shows the theoretical increased costs of bug fixes over time.[41] Memory safety enforces correctness at compile time, lowering the cost of maintaining software by keeping memory bugs out of production at all. Companies like Nvidia, Amazon, and Google have seen that impact to operations in their migrations to memory-safe languages.

[41] Penny Grubb and Armstrong A. Takang, *Software Maintenance: Concepts and Practice*, 2nd ed. (World Scientific Publishing, 2003), 26.

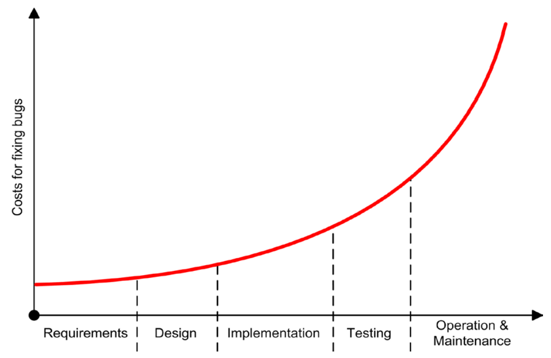


Figure 2. Cost of fixing bugs over the software development life cycle.

Fixing bugs and security vulnerabilities is necessary corrective maintenance, and sometimes engineers introduce new, unintended problems called "regressions" when implementing them. There is a nontrivial likelihood that a well-intentioned code fix will have an adverse impact because of the complexity of modern systems and lack of institutional knowledge due to high engineering turnover.[42] Memory safety prevents engineers from introducing new memory bugs, which decreases the frequency of regressions and lowers the cost of bug fixes.

The Nvidia Offense Security Research team compared code bugs in their solutions built with unsafe (C/C++) and memory-safe (SPARK) programming languages. They did side-by-side comparisons of safe and unsafe for root of trust and resource management, operating systems, and boot control. **Nvidia found 71–78 percent fewer bugs in memory-safe implementations of their memory-unsafe software**, and on average, 54 percent of the bugs identified in the unsafe code were memory-safety bugs.[43] Using a memory-safe programming language dramatically decreased the number of bugs in their software, delivering a similarly dramatic decrease in the cost of operating that software.

Adding New Features to Support Evolving User Needs

Finally, modifying Rust is less risky than memory-unsafe languages, generating tremendous savings over its lifetime. Billions of lines of code in production today were written decades ago,[44] and the developers operating and maintaining those applications are not their original authors. The technology industry has one of the highest employee turnover rates at 12.9 percent, which is more than 20 percent higher than the average for all industries.[45] For a ten-person engineering team, that means that half the team building a new software solution will be gone within four years, and none of the original authors will still be working on the

[42] Greg Lewis and Joseph Soroñgon, "Industries With the Highest (and Lowest) Turnover Rates," LinkedIn Talent Blog, Aug. 11, 2022, https://www.linkedin.com/business/talent/blog/talent-strategy/industries-with-the-highest-turnover-rates.
[43] Zabrocki and Tereshkin, supra note 36.
[44] Owen Hughes, "This Old Programming Language Is Much More Important Than You Might Expect. Here's Why," *ZDNet*, Feb. 9, 2022, https://www.zdnet.com/article/programming-languages-how-much-cobol-code-is-out-there-the-answer-might-surprise-you/.
[45] Lewis and Soroñgon, supra note 42.

application in eight years. As a result, the engineers maintaining software frequently do not have a deep understanding of how the software was designed, and they will approach the discovery process for understanding the behavior of the code differently.

The safety net provided by a memory-safe language makes engineering teams more agile, because code changes are less risky. While the Rust compiler's strict adherence to correctness might have slowed the initial development of software, the engineers tackling maintenance of that system will reap its benefits. The compiler will not overlook memory errors, and the engineers' experimentation will yield feedback at compile time rather than later during runtime testing. That feedback loop is faster and less expensive, improving the productivity of engineers maintaining and operating software written in Rust.

Rust is also the easiest programming language to sight-read. Engineers reading new code are like musicians reading unfamiliar sheet music. There are always recognizable elements, but the theme, pace, and key may be outside of the player's experience. In software, those unfamiliar elements can take a developer through a complicated maze of dependencies and logic trees, and Rust makes the trail of logic in a program easier to follow. Researchers have concluded that Rust has a significantly lower cognitive complexity than C, C++, Python, JavaScript, and TypeScript (all languages studied), "meaning that [Rust] can guarantee the highest understandability of source code compared to all others."[46] As a result, software maintainers can understand unfamiliar Rust code far more quickly than code written in many other popular languages.

The improved understandability of Rust as well as the reduced risk of regression make new features less expensive and disruptive for an engineering team. Many legacy systems are stuck in time because of lost opportunities to make noncritical improvements. Features, fixes, and upgrades are often low impact individually, and for complex systems written in memory-unsafe programming languages, they are not worth the risk of regressions that can introduce new, invisible security vulnerabilities. Using a memory-safe language reduces the cost of lost opportunities for improvements to software over its lifetime.

TAKEAWAYS FOR POLICYMAKERS

The Rust programming language fills a substantial gap in memory-safe language solutions, making memory safety possible in far more domains. In addition to improving cybersecurity for technology manufacturers, memory safety lowers the total cost of maintaining software over its lifetime, delivering real value to both consumers and producers. Despite those advantages, Rust may always be a niche solution on the bleeding edge of tech because of the lock-in memory-unsafe languages have achieved and the market friction inherent in Rust's design and support. That lock-in means that malicious actors will continue to exploit vulnerabilities that would be prevented with memory safety—vulnerabilities like buffer overflows, use-after-free, and out-of-

[46] Luca Ardito, Luca Barbato, Riccardo Coppola, and Michele Valsesia, "Evaluation of Rust Code Verbosity, Understandability and Complexity," *PeerJ Computer Science* 7 (2021). doi: 10.7717/peerj-cs.406.

bounds reads and writes that enable threats like the 1988 Morris worm, the 2016 Heartbleed bug, the 2016 Trident attack, and the 2017 WannaCry attack.

For many resource-constrained technologies, like mobile phones and the networks that connect them, Rust is the only viable memory-safe language available today, but Rust's immaturity introduces new risks for technology manufacturers. Rust has (a) limits to the scope of its memory safety, (b) missing audits and alerts for code that has disabled memory safety,[47] (c) missing standard security tools for memory-unsafe Rust code,[48] and (d) an unusually large number of third-party dependencies.[49] It is not true that all Rust code is memory safe, and we have a lot more work to do before it is.

Like most popular programming languages, Rust is not a single product. The Rust programming language is a collection of open source projects built and maintained by thousands of people over more than a decade, and **like all open source software, Rust is available "as is" with no warranty.** There is no authority responsible for the memory-safety claims of Rust nor liable for its failures. That is true for all open source software, and some of the challenges with Rust's stability and maturity are common across open source projects.

Today, open source delivers foundational code for almost every technology, as community freeware has penetrated every domain. Aerospace, automotive, mobile phones, "internet of things," e-commerce, artificial intelligence, health care, virtual reality—they are all open source.[50] Open source projects are like Lego building blocks, and engineers create consumer technology by putting these blocks together in different ways to create unique solutions. With few exceptions, open source software is not owned or maintained by any single legal entity. Open source offers no maintenance contract nor responsible authority. Nothing is guaranteed nor warrantied.

Public policy can have historic economic impact by addressing risks to critical open source projects like memory-safe languages as well as the market hurdles that stall adoption of new security solutions like Rust. This paper recommends

- an addition to the critical infrastructure information technology sector,
- a cloud computing tax to fund critical U.S. cyber defense,
- U.S.-sponsored governance for emerging cybersecurity solutions like Rust, and
- a U.S.-sponsored open source library verification service.

[47] Steve Klabnik and Carol Nichols, "Unsafe Rust," The Rust Programming Language, https://doc.rust-lang.org/book/ch19-01-unsafe-rust.html.
[48] Joe Sible and David Svoboda, "Rust Software Security: A Current State Assessment," Carnegie Mellon University Software Engineering Institute, Dec. 12, 2022, https://insights.sei.cmu.edu/blog/rust-software-security-a-current-state-assessment/.
[49] Sergio De Simone, "Static Analyzer Rudra Found Over 200 Memory Safety Issues in Rust Crates," *InfoQ*, Nov. 13, 2021, https://www.infoq.com/news/2021/11/rudra-rust-safety/.
[50] Bals, supra note 27.

Addition to the Critical Infrastructure Information Technology Sector

Programming languages and hardware (physical machines) are the roots of all technology, and the design, implementation, and management of programming languages is done largely by anonymous open source community volunteers all over the world. **We are connected across geographies, politics, and socioeconomic boundaries not just by the technologies we use but also in the collaborative creation of those technologies**, and policymakers should incorporate an understanding of those relationships into both foreign and domestic policy.

If you look closely at the long agreements that come with the technology you're using today, you'll see names of open source engineers like Daniel Stenberg where a company's ought to be. Daniel's name and software project (curl) are listed in the copyright for everything from *Grand Theft Auto* to Spotify to Volkswagen minivans.[51] Moreover, Daniel's work depends on Sean McArthur's, and Sean McArthur's work depends on Vadim Petrochenkov's, and so on—a global web of descendants of the early internet trailblazers.

That global community is supported by companies and governments all over the world, including substantial investment from China. In fact, the Chinese company Huawei holds more than 450 key positions within 800 standards organizations, industry alliances, open source communities, and academic associations,[52] including board seats on open source foundations like Linux Foundation, Rust Foundation, and OpenSSF. The Chinese company ByteDance holds a board seat at the Apache Software Foundation, and both ByteDance and Kuaishou Group are members of the Open Invention Network, the largest patent non-aggression consortium worldwide. Futurewei, the U.S.-based research and development subsidiary of Huawei, directly employs roughly half of the full-time paid maintainers of the Rust project. U.S. restrictions on China's access to advanced hardware[53] are inconsistent with our significant dependence on Chinese funding for open source projects like the Rust programming language.

Early Rust adoption is focused on security-sensitive domains like mission critical systems, infrastructure services, and operating systems, while the Rust toolchain and ecosystem have significant security vulnerabilities with high likelihood of attack and high severity impact that the community is working to address.[54] That community is an international partnership with the most substantial contributions of time, talent, and money coming from both the United States and China, and that is challenging to navigate in the current political climate. To be sure, the way forward must be cognizant of ongoing strategic adversarial imperatives, but at the same time, technical collaborations with Chinese organizations can be meaningful. The U.S.

[51] Daniel Stenberg, "Screenshotted Curl Credits," Daniel Stenberg blog, Oct. 3, 2016, https://daniel.haxx.se/blog/2016/10/03/screenshotted-curl-credits/.
[52] "Openness, Collaboration, and Shared Success," Huawei, https://www.huawei.com/en/corporate-information/openness-collaboration-and-shared-success.
[53] Boak, supra note 25.
[54] Rust Foundation, "Security Initiative Report," Feb. 15, 2024, https://foundation.rust-lang.org/static/publications/security-initiative-report-february-2024.pdf.

government must take responsibility for managing the inherent risks, so that these open, global communities can continue to focus on pioneering technology advancements.

U.S. government agencies prevent, deter, and mitigate risks to sectors identified by the Cybersecurity and Infrastructure Security Agency (CISA) as "critical infrastructure." The information technology (IT) sector is one of the sixteen critical infrastructure sectors managed by the U.S. government today,[55] and CISA's IT sector protection plan covers six critical functions. These include core functions like the Domain Name System (DNS), internet routing infrastructure, and communication services. Programming languages are also a technology vital to our national economic security, and they should be added as the seventh critical function of the IT sector. The IT sector protection plan for programming language risks should include mechanisms to identify the most critical languages, continuously evaluate the security and stability of those languages, provide regular public reports on weaknesses identified and mitigated, and respond to vulnerability reports and support requests from language stewards like the Rust Foundation.

Cloud Computing Tax for U.S. Cyber Defense

Today's technology stands on a foundation of public goods created by passionate volunteers, and our global security depends not on "the industry" as some security leaders have asserted, but on the quality, stability, and security of the open source building blocks the tech industry uses. The U.S. government must take real responsibility for defending the American public from the outsized risks of sharing critical public goods that are the foundation of our economic infrastructure, with state actors whose "multi-pronged assault on our national and economic security" they believe to be "the defining threat of our generation."[56] Congress and the president cannot delegate the consumer safety of public goods to private institutions like big tech, but they can demand that big tech, as the primary beneficiary of open source software, be held accountable for funding the security of that work without which their own products would cost 3.5 times more to build.[57]

In 1956, Congress passed the Interstate and Defense Highways Act, creating a fund that uses fuel taxes to pay for critical American roads and bridges, and today, our national economy demands an equally bold investment in our digital networks. Following the Interstate Act's model that uses a consumption tax to pay for the interstate infrastructure on which consumers depend, Congress should act quickly to pass an Internet Defense Act that creates a federal cloud computing tax to fund a new Open Source Trust and increase funding for existing national

[55] U.S. Cybersecurity and Infrastructure Security Agency, "Critical Infrastructure Sectors," https://www.cisa.gov/topics/critical-infrastructure-security-and-resilience/critical-infrastructure-sectors.
[56] U.S. Federal Bureau of Investigation, "Director Wray's Opening Statement to the House Select Committee on the Strategic Competition Between the United States and the Chinese Communist Party," Jan. 31, 2024, https://www.fbi.gov/news/speeches/director-wrays-opening-statement-to-the-house-select-committee-on-the-chinese-communist-party.
[57] Manuel Hoffman, Frank Nagle, and Yanuo Zhou, "The Value of Open Source Software," *Harvard Business School Working Paper* No. 24-038, January 2024, https://www.hbs.edu/faculty/Pages/download.aspx?name=24-038.pdf.

security organizations and partners like the National Institute of Standards and Technology (NIST), CISA, and the Carnegie Mellon University Software Engineering Institute (CMU SEI).

While twenty-four states currently tax the sales of software-as-a-service (SaaS),[58] there is no state or federal sales tax on infrastructure-as-a-service (IaaS). That means that when Meta pays Amazon Web Services for cloud compute services like Amazon EC2, no sales tax is due. At the same time, researchers estimate that the cost of creating technologies like Amazon EC2 without open source software would be 3.5 times higher than it is today.[59] Introducing a cloud sales tax to fund improvements to the security and stability of open source software used to build the cloud makes cloud computing more secure and improves the security and stability of open source for all technology. A cloud computing tax is long overdue, and it must be collected to secure the software supply chain for American consumers.

The Interstate and Defense Highways Act introduced a federal fuel tax that was 18.4 cents per gallon in 2023,[60] while the average price of gasoline was $3.634 per gallon,[61] making the effective gasoline tax a little more than 5 percent. The biggest (American) cloud providers generated $270 billion in revenue that year,[62] and a comparable federal cloud computing tax of 5 percent would have produced $13.5 billion for cybersecurity in 2023 alone. That is the order of magnitude that is both possible and required to protect America's critical infrastructure.

A cloud sales tax would put the cost of securing open source for U.S. economic stability on the companies that have profited the most from open source software—its biggest consumers. The Open Source Trust can offer financial support to open source communities, allow for more free-flowing exploration of our technology frontier, and close a gaping hole in America's economic stability. In the White House's own words, "Government's role is to protect its own systems [and] to ensure private entities, particularly critical infrastructure, are protecting their systems[.]"[63] Technology companies rely on open source as a public good, and policymakers owe it to American consumers to ensure the security and resilience of that public good.

U.S. Governance for Emerging Cybersecurity Solutions

We cannot make memory-safe languages an industry standard without mainstream adoption, and today, Rust is the most prominent memory-safe language for performance sensitive

[58] "Introduction to SaaS taxability in the US," Stripe, https://stripe.com/guides/introduction-to-saas-taxability-in-the-us.
[59] Hoffman, et al., supra note 57.
[60] U.S. Department of Transportation, "When Did the Federal Government Begin Collecting the Gas Tax?" Highway History, https://www.fhwa.dot.gov/infrastructure/gastax.cfm.
[61] U.S. Energy Information Administration, "U.S. All Grades All Formulations Retail Gasoline Prices (Dollars per Gallon)," Petroleum & Other Liquids, https://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=pet&s=emm_epm0_pte_nus_dpg&f=m.
[62] Mark Haranas, "Cloud Market-Share Q4 2023 Results: AWS Falls as Microsoft Grows," *CRN*, Feb. 2, 2024, https://www.crn.com/news/cloud/2024/cloud-market-share-q4-2023-results-aws-falls-as-microsoft-grows.
[63] The White House, *National Cybersecurity Strategy*, supra note 8.

domains. At the same time, Rust has big, open challenges that require strong product, engineering, and program leadership, but Rust's decentralized governance and individualized priorities prevent the community from organizing the big improvements needed to break into the mainstream market. Rust must leap across the market chasm pictured in Figure 3 to transition from innovators and visionaries on the leading edge of technology to pragmatists in the mainstream market. Pragmatists, or the early majority, are not adventurous founders and innovators. These are the folks that commit and stay the course, and their goal is to make "incremental, measurable, predictable progress."[64] Market pragmatists invest in proven technologies offered at reduced prices, and they are the gateway to the entire mainstream market. New technologies remain niche solutions used by a few tech enthusiasts until they are accepted and adopted by market pragmatists.

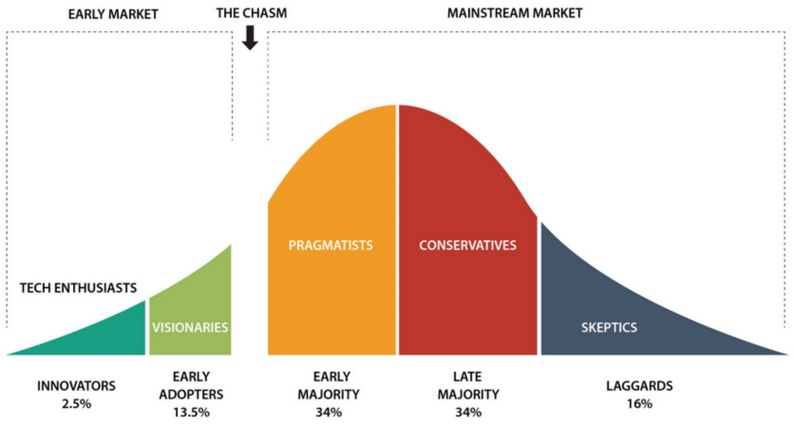


Figure 3. Breaking into the mainstream market.

Rust was built by more than nine thousand volunteers. While many successful open source projects (e.g., Linux) have leaders with decision-making authority, other open source communities like Rust have no comparable leadership. The fluid development and organizational slack permitted by Rust sponsors and employers incubates innovation that's not possible in more structured organizations,[65] but that same lack of governance inhibits the community's ability to deliver results efficiently and effectively. As a result, large initiatives that require more than a couple of people to collaborate and commit are not achievable, and external governance intervention will be necessary to close the gaps in Rust for mainstream users.

Taking Rust across the chasm from early adopters to the mainstream market requires (a) credible testimonies on the costs and returns of Rust investments and (b) reductions in costs and risks of using Rust without losing benefits. The mainstream market relies on success stories from trusted references. It seeks evidence that new technologies can succeed, and too many early adopters of Rust are keeping the details of their migrations private. The mainstream market is not interested in whether Amazon, Google, Microsoft, and Meta are using Rust—

[64] Geoffrey A. Moore, *Crossing the Chasm,* 3rd ed. (HarperCollins, 2014), 55.
[65] Rogers, supra note 19, p. 412.

mainstream market leaders and engineers want to know how much those companies are paying to adopt Rust and what the return on that investment has been. There are private anecdotes of Rust code outperforming legacy C applications because engineers felt more confident optimizing Rust. There are also private buyer-beware tales of Rust engineering teams blocked on performance regressions for days because the tools available to inspect Rust performance issues are inadequate.

Programs that create actionable guidance and referenceable case studies for memory-safety adoption have a big effect, and incentivizing early adopters to share their Rust migration successes and challenges publicly would materially influence memory-safety adoption. Technology companies announcing a few million dollars' investment in their internal projects, maintainer sponsorships, or foundation donations have little impact beyond the direct recipients of those funds. **Widespread memory-safety adoption requires credible observability.**

In December 2023, CISA, the National Security Agency, and the FBI partnered with similar agencies from four other countries to author "The Case for Memory Safe Roadmaps" that "urge[s] software manufacturers to create and publish memory safe roadmaps that detail how they will eliminate memory safety vulnerabilities in their products."[66] Publishing documents that articulate a plan for Rust migration as part of a memory-safety roadmap would substantially improve the observability of existing Rust adoption and spur near peers to begin their own planning, but this is a giant first step. There are incremental successes that could make the larger effort of a roadmap more achievable while delivering earlier observability.

The roadmap guidance developed by U.S. government agencies and international partners offers very few insights to technology manufacturers interested in pursuing a memory-safety roadmap, and the suggestions provided are ambiguous and expensive. The roadmap guidance for prioritization includes advice like "start with new and smaller projects" and "prioritize security critical code," and guidance for planning advises steps like "ensure teams have access to training" and "create a staffing pipeline." A public-private partnership effort to build an actionable cookbook for memory-safety migration would be a better first step than urging technology manufacturers to use the one available today.

CISA should partner with early Rust adopters to identify their insights, costs, and wins and visibly incorporate that data into the roadmap guidance. The guidance from that exercise would reduce complexity, increase perceived relative advantage, and improve confidence across the industry. The objectives of this guidebook would be to (a) describe clearly and unambiguously how to approach a migration, (b) establish realistic expectations for upfront costs and risks, (c) identify organizations, initiatives, and alliances available for support, and (d) provide a calculator for the long-term cost savings that technology manufacturers can expect after their migration to memory safety. CISA should lead an initiative to create this cookbook for memory-safety

[66] U.S. Cybersecurity and Infrastructure Security Agency, "The Case for Memory Safe Roadmaps," Dec. 6, 2023, https://www.cisa.gov/resources-tools/resources/case-memory-safe-roadmaps.

migration starting with Rust, where there is little institutional knowledge available today, and this work should be funded by the Open Source Trust.

A deep dive into memory-safety migrations will also make the limitations and risks of memory-safe languages like Rust clear to CISA, giving them the data to lead more effective governance of critical tools that CISA believes can address a large class of cybersecurity vulnerabilities. For example, the Rust compiler, which enforces Rust's safety guarantees, has limitations that have an outsized impact on the effectiveness of migrations, where there is likely to always be some residual memory-unsafe code in a system. The Rust compiler checks the memory safety of an engineer's code before transforming it into an executable program, but when Rust code is interfacing with unsafe code in languages like C and C++, the compiler does not have enough information to classify behavior as memory safe. Rust includes a special keyword ("unsafe") that an engineer can use to tag these kinds of implementations, and the Rust compiler will skip memory-safety validations on the tagged code.

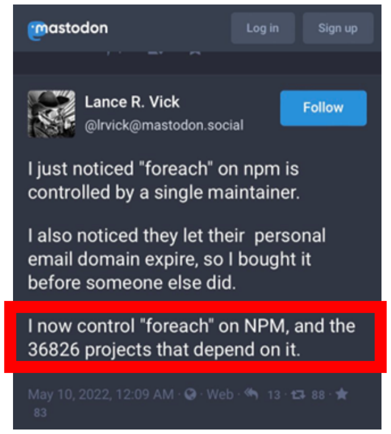
Since this is comparable to taking the memory-safety properties out of Rust, software engineers are expected to limit the use of "unsafe Rust" to the smallest possible scope. However, anecdotal reports of faster software performance with "unsafe Rust" have incentivized using it more widely and introduced additional risk. Some Rust libraries use "unsafe Rust" more broadly than necessary, reducing the memory safety of their Rust code. "Unsafe Rust" is unavoidable today, but engineers need education and tools to know when to use it and how to mitigate the risks "unsafe Rust" introduces.

As a result of the gaps in Rust's memory-safety and analysis tools, Carnegie Mellon University Software Engineering Institute researchers "categorize Rust as a *safer* language, rather than a *safe* language, because the safety Rust provides is limited."[67] More mature memory-safe and memory-unsafe languages are evaluated for cybersecurity vulnerabilities by software engineers using static and dynamic analysis tools, and there are only experimental, proof-of-concept analysis tools available for Rust today.[68] CMU SEI should receive Open Source Trust funding to continue their research and development and (a) reduce the limitations of the Rust compiler, (b) audit the Rust compiler's correctness in assessing the memory safety of Rust code, and (c) develop both static and dynamic analysis tools for safe and unsafe Rust.

Rust is also susceptible to open source security risks. Two years ago, the author of the social media post shown in Figure 4 described taking control of a popular JavaScript library used by more than thirty-six thousand software programs to illustrate a common open source security threat that leaves most programming languages vulnerable.[69] Threat models for build tools, package managers, and compilers are similar across languages, and the tech industry is long

[67] Sible and Svoboda, supra note 48.
[68] Garret Wassermann and David Svoboda, "Rust Vulnerability Analysis and Maturity Challenges," Carnegie Mellon University Software Engineering Institute, Jan. 23, 2023, https://insights.sei.cmu.edu/blog/rust-vulnerability-analysis-and-maturity-challenges/.
[69] Florian Roth [@cyb3rops], "Pwn the world mastodon.social/@lrvick/108274," Twitter, May 10, 2022, https://twitter.com/cyb3rops/status/1523979837769142273.

overdue for a systematic approach to mitigating inherent weaknesses in their governance that make threats like this common.



Lance R. Vick
@lrvick@mastodon.social

I just noticed "foreach" on npm is controlled by a single maintainer.

I also noticed they let their personal email domain expire, so I bought it before someone else did.

I now control "foreach" on NPM, and the 36826 projects that depend on it.

May 10, 2022, 12:09 AM · Web · 13 · 88 · 83

Figure 4. An illustration of open source security risks.

When this weakness was discussed on social media in 2022, thousands of language libraries were controlled by expired domains, including more than one thousand Rust libraries (crates). Individuals decide on a case-by-case basis whether to react to threats like domain expirations, and they can implement mitigation measures without oversight or transparency, ignore the risk these threats pose to the entire industry, or take advantage of them for malicious purposes. A trusted, publicly funded organization must address the collective action challenge of programming languages like Rust by establishing funded standards for popular and promising memory-safe language toolchains.

CISA has partnered with the OpenSSF working group to outline cybersecurity assessment dimensions for open source package managers.[70] The principles and scoring system the team has come up with are a good start, and minimum standards for those scores that protect consumers is what needs to follow. CISA should receive additional Open Source Trust funding to support rapid, in-depth development of standards across package repositories, compilers, and build tools, along with program management capabilities to collaborate with the stewards of those tools, like the Rust Foundation, and to incentivize the work needed to satisfy CISA standards.

U.S. Open source Library Verification Service

Mobile devices, operating systems, and cloud infrastructure can prevent more than half of their security vulnerabilities by migrating to a memory-safe programming language, and early efforts to make progress using Rust for those use cases have driven an incredible rate of growth in the Rust ecosystem and maturity over the past five years. In 2019, there were just 400,000

[70] Jack Cable and Zach Steindler, "Principles for Package Repository Security," OpenSSF, February 2024, https://repos.openssf.org/principles-for-package-repository-security.

engineers worldwide using Rust,[71] and there were only three paid maintainers working on the compiler and standard library. Everything else was accomplished by open source community volunteers. Engineers who wanted to build with Rust were struggling to learn the language and make it work for their needs. Critical pieces of the Rust ecosystem were so incomplete that they were unusable in production systems, and there were no reliable estimates of when that would change. Only basic educational resources were available, and very few experts offered mentoring and coaching.

In the five years since then, the Rust developer community has exploded nearly 800 percent to 3.5 million engineers.[72] The Rust Foundation was launched in 2020 and is supported by more than forty different corporations, and there are at least thirty full-time paid maintainers and staff working on the Rust compiler, package manager, and standard library. More recently, AdaCore announced plans to offer a certified and qualified Rust toolchain, contracts for service-level agreements, and lifetime support for backported fixes. There are also a dozen small businesses all over the globe offering consulting and training services for Rust development. Rust has come a long way, but the tech industry is just taking its first steps on a multi-decade journey to memory-safe software everywhere.

There are still substantial gaps in Rust that add cost, complexity, and risk to building with it, and the mainstream market requires a whole product solution that's easy to identify and trust. Mainstream companies will not dig through thousands of tiny Rust libraries to find what they need. They don't want to have to review hundreds of strangers' code for "unsafe Rust" and evaluate its quality and security. They don't want to have to worry about the security and stability of updates to the Rust libraries they use. **Mainstream companies will invest in technologies they trust to deliver.** Programming languages with low risk and complexity can be trusted by technology manufacturers, and that can be achieved only with safe, stable core language tools and easy verification of safe, supported language libraries.

Like most popular programming languages, Rust is a collection of open source projects built and maintained by thousands of people over more than a decade, and like all open source software, the Rust language and ecosystem are provided "as is," with no warranty. Memory safety does not absolve technology manufacturers' responsibility to verify the security of third-party open source dependencies, and the combination of Rust's uniquely small libraries and immature implementations exacerbates that challenge. Rust basic tools (like http and serialization) are spread across many more independent libraries than in more mature languages, and Rust engineers typically require a few hundred of these libraries for a single software project.[73] That's far more open source libraries and authors to evaluate and manage than the typical Python

[71] Michael Carraz, et al., "State of the Developer Nation, 17th Edition," SlashData, Oct. 22, 2019, https://www.developernation.net/resources/reports/state-of-the-developer-nation-17th-q2-2019/.
[72] Liam Dodd, et al., "State of the Developer Nation, 25th Edition," SlashData, Nov. 23, 2023, https://www.developernation.net/resources/reports/state-of-the-developer-nation-25th-edition-q3-20231/.
[73] "Motivation," Cargo Vet, Mozilla, https://mozilla.github.io/cargo-vet/motivation.html.

program with only thirty-five dependencies.[74] At the same time, fewer than 30 percent of the most commonly used Rust libraries have a stable version with long-term (volunteer) support available, and that ratio is growing by less than 2 percent annually. The additional complexity of the Rust language demands that engineers do far more exploration, validation, and verification than mature programming languages, and without a central, trusted authority, every technology manufacturer must replicate that work.

While the Rust Foundation includes the entire Rust ecosystem in its bylaws, the core Rust project is the only community represented on its board of directors, and it is the primary recipient of all funding. None of the authors or maintainers of the 145,000 Rust language libraries is represented or supported, and no organization offers a library verification service. Private companies have failed to produce a market solution for Rust language entropy and immaturity, because the demand for Rust is not big enough to support it today. Corporations offering solutions for everything from open source analysis tools to package management policies are waiting for Rust to cross the chasm into the mainstream before extending their existing products to support Rust. At the same time, like any open source software, Rust is susceptible to security and quality problems like malicious packages and negligence, and the Rust Foundation's security threat model gives high ratings to both severity and likelihood of a malicious library attack.[75]

An initial investment needs to be made to protect technology manufacturers adopting Rust now, even while it is not profitable for a corporation to do so, and the nonprofit Internet Security Research Group (ISRG) has been working since 2020 to drive adoption of memory safety by making some progress in this area. While ISRG has successfully captured more than ten sponsors for the work to sustain and improve memory-safe software, they have barely scratched the surface of what needs to be done. At the same time, the adoption of the work completed by ISRG has been disappointing.

ISRG's multiyear effort to transition curl, a prolific open source solution, to memory-safe http[76] is still unused years after the option was made available, and the software's maintainer will likely remove support for memory safety due to the lack of interest from their users.[77] A nonprofit like ISRG must use Open Source Trust funds to create a complete library verification service that identifies, maintains, and hosts verified and validated memory-safe language libraries that provide standard software solutions (like encryption and serialization), and the U.S government must leverage tools like NIST's secure software standards and frameworks to encourage adoption of those memory-safe solutions.

[74] "Python Security Insights," Snyk Report, September 2021, https://go.snyk.io/rs/677-THP-415/images/Python Insight Report.pdf.
[75] Rust Foundation, supra note 54.
[76] Josh Aas, "Memory Safe 'curl' for a More Secure Internet," Internet Security Research Group, Oct. 9, 2020, https://www.abetterinternet.org/post/memory-safe-curl/.
[77] Daniel Stenberg, "Hyper, Is It Worth It?", curl, April 16, 2024, https://curl.se/mail/lib-2024-04/0021.html.

CLOUD ANALOGY—WE CAN DO IT

Changing fundamentals like software architectures or programming languages is expensive, but it can be done and has been done before. Researchers estimate the cost of the Y2K remediation effort to have been between $300 and $600 billion,[78] and cloud computing is expected to increase 20.4 percent year over year to $678.8 billion in 2024.[79] The transition to cloud has been a massive migration effort for the technology industry.

To put the size of that migration in perspective, between 2010 and 2022, energy consumption by cloud data centers increased 500 percent, while energy consumption by private data centers decreased 75 percent.[80] That's the result of migrating storage and compute from those legacy data centers to the cloud driven by lower costs, shared responsibility, and greater flexibility achieved by replacing legacy systems with modern implementations.

Cloud migration succeeded slowly at first, and early adopters were small and medium businesses looking to lower the cost of entry into new markets and reduce business risks associated with spikes in usage (like over- or underspending on infrastructure). Early adopters like Netflix, Uber, and Airbnb didn't have huge legacy systems that make cloud adoption expensive, so they were able to get into the cloud quickly and cheaply. Larger, older companies followed because of the business agility those early cloud-based companies achieved, making the cloud the gold standard for software operations. As recently as 2016, leaders at top cloud companies were asking whether large enterprises like multinational banks should or could transition to the cloud, and by 2021, "more than 90 percent of banks [surveyed by the American Bankers Association] stated that they maintain at least some data, applications, or operations in the cloud."[81]

The cloud transition was boosted by large, early corporate investments in the U.S. government as a customer. The first public cloud vendor, Amazon Web Services, launched a dedicated government region in 2011,[82] many years before there was enough government demand for cloud computing to make that investment profitable. Cloud providers believed that having U.S. government agencies trust their data and services to the cloud makes large enterprises comfortable doing the same, and historically, that has been true. Financial institutions, health

[78] "Y2K", Smithsonian, https://www.si.edu/spotlight/y2k.

[79] Gartner, "Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach $679 Billion in 2024," Nov. 13, 2023, https://www.gartner.com/en/newsroom/press-releases/11-13-2023-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-679-billion-in-20240.

[80] International Energy Agency, "Global Data Centre Energy Demand by Data Centre Type, 2010-2022," June 2, 2020, https://www.iea.org/data-and-statistics/charts/global-data-centre-energy-demand-by-data-centre-type-2010-2022.

[81] U.S. Department of the Treasury, "The Financial Services Sector's Adoption of Cloud Services," Feb. 8, 2023, https://home.treasury.gov/system/files/136/Treasury-Cloud-Report.pdf.

[82] Dave Levy, "10 Years of Government Cloud Innovation With AWS GovCloud (US)," Amazon Web Services, Sept. 28, 2021, https://aws.amazon.com/blogs/publicsector/10-years-of-government-cloud-innovation-aws-govcloud-us/.

care providers, and other critical infrastructure reference the U.S. government's trust in cloud computing security and reliability to satisfy their own requirements for cloud adoption. Cloud migration is moving comfortably into the mainstream market this year, and Gartner predicts that "by 2027, more than 70% of enterprises will use industry cloud platforms to accelerate their business initiatives, up from less than 15% in 2023."[83] Investments like those made in cloud computing could substantially speed up the memory-safe transition in a similar way.

CONCLUSION

Today, many technology manufacturers approach software development with short-term tactics and often end up with long-term ownership of subpar products. Corporations are underwriting unstable, fancy features with a line of credit, and customers are saddled with the security and reliability consequences. In 2015, Gartner estimated the total global IT debt from technology manufacturers "taking shortcuts, using basic techniques, not considering long-term consequences […], and delaying the upgrade of infrastructure" at $1 trillion, and Deloitte described that cost as an "accumulation of financial liabilities."[84] Adopting a memory-safe-by-default approach to software development would dramatically improve customers' experience and security, slow the accumulation of new technical debt, and lower the cost of ownership for technology manufacturers for the decades they are likely to operate and maintain business-critical software.[85]

Most modern programming languages are memory-safe, and they share a common memory-safety design[86] that boosts speed-to-market at the cost of slower and more resource expensive software. For example, JavaScript's combination of consistency across web browsers and support for dynamic features makes it the most widely used programming language, with 22.5 million active engineers (58 percent of all software engineers).[87] Similarly, Python's dynamic typing and simplified abstractions make it relatively easy to use and accessible to less traditional software engineers like data scientists and security specialists. As a result, Python is in third place, with 16.9 million active developers (43 percent of all software engineers).[88]

The productivity gains of legacy memory-safe languages have been a huge windfall for cybersecurity, because they tap into the tech sector's intense pressure to get new products to market quickly. In an increasing returns market like technology, the first product to get ahead

[83] Gartner, supra note 79.

[84] Ranjan Sinha, Mohammed Arshad Hussain, Adib Ibrahim, et al., "Could the Cloud Be the Solution to Addressing Technical Debt?" Deloitte, 2019, https://www2.deloitte.com/content/dam/Deloitte/xe/Documents/technology/me-consulting_technical-debt.pdf.

[85] Phil Murphy, "New Jersey Needs COBOL Programmers," YouTube, uploaded by Joseph Steinberg, April 4, 2020, https://www.youtube.com/watch?v=HSVgHlSTPYQ.

[86] Zixian Cai, Stephen M. Blackburn, Michael D. Bond, and Martin Maas, "Distilling the Real Cost of Production Garbage Collectors," *IEEE International Symposium on Performance Analysis of Systems and Software*, 2022, pp. 46–57. doi: 10.1109/ISPASS55109.2022.00005.

[87] Dodd, supra note 72.

[88] Ibid.

wins momentum that will move it further ahead. The technologies that can afford the additional overhead of those older, resource-expensive, memory-safe languages have leveraged them to build new features faster. At the same time, resource-constrained technologies like phones, automobiles, and networks have primarily stayed locked in to memory-unsafe languages that deliver the performance and optimizations those technologies require.

To fix this over-reliance on legacy memory-unsafe languages, several things are needed. The current U.S. guidance on Chinese technical partnerships is contradictory and makes success impossible for all stakeholders. The American tech community needs clear guidance and governance of open collaborations with China. Government intervention is also required to jump-start broad market adoption of memory safety everywhere. The Internet Defense Act must establish a cloud computing tax to fund improvements to observability, governance, and complexity for emerging memory-safe languages like Rust through the Open Source Trust. Public investments can reduce risks to memory-safety adoption by providing:

- an addition to the critical infrastructure information technology sector,
- a cloud computing tax to fund critical U.S. cyber defense,
- U.S.-sponsored governance for emerging cybersecurity solutions like Rust, and
- a U.S.-sponsored open source library verification service.

As CISA Director Jen Easterly said in her recent congressional testimony, "[C]ybersecurity is national security" and "cyber risk is business risk."[89] The cadence of cyberattacks is increasing, and technology manufacturers that fail to adopt cybersecurity best practices like memory safety are putting their shareholders and customers at risk. Technology manufacturers must proactively partner with U.S. government agencies to drive progress for our shared responsibility to make all software memory-safe-by-default. Secure and stable memory-safe programming languages can secure our critical infrastructure, protecting everything from our natural resources to our national economy. Investing in Rust is good for the security of our nation, and it's good for business.

[89] "Select Committee Hearing on China's Cyber Threat to the U.S.," C-SPAN, clipped by Jen Easterly, Jan. 31, 2024, https://www.c-span.org/video/?c5104695/user-clip-jen-easterly.