BACK TO THE BUILDING BLOCKS:

A PATH TOWARD SECURE AND MEASURABLE SOFTWARE

FEBRUARY 2024

THE WHITE HOUSE
WASHINGTON

Table of Contents

ABSTRACT

President Biden's National Cybersecurity Strategy outlines two fundamental shifts: the need to both rebalance the responsibility to defend cyberspace and realign incentives to favor long-term cybersecurity investments. In this report, the case is made that the technical community is well-positioned to drive progress on both strategic goals. First, in order to reduce memory safety vulnerabilities at scale, creators of software and hardware can better secure the building blocks of cyberspace. This report focuses on the programming language as a primary building block, and explores hardware architecture and formal methods as complementary approaches to achieve similar outcomes. Second, in order to establish accurate cybersecurity quality metrics, advances can be made to address the hard and complex research problem of software measurability. This report explores how such metrics can shift market forces to improve cybersecurity quality across the ecosystem.

PART I: INTRODUCTION

Users of software and hardware products are consistently placed in the untenable position of reacting to cyber emergencies. Responding on a crisis-by-crisis basis often leaves them on their heels, and those securing systems on the front lines should not be expected to bear the full weight of this burden. The intense reactive posture demanded by the current status quo reduces defenders' ability to predict and prepare for the next wave of incoming attacks.

This posture stems from the fact that mitigating known software vulnerabilities is a complex systems problem and the current ecosystem does not sufficiently incentivize the investments required to secure the foundations of cyberspace. Since 2021, the Biden-Harris Administration has taken major action, starting with Executive Order 14028 on Improving the Nation's Cybersecurity, to drive the ecosystem to patch known classes of vulnerabilities through secure software development practices across the supply chain. Continuing to encourage both the government and the private sector to do this can have an outsized impact on improving the Nation's cybersecurity.

However, even if every known vulnerability were to be fixed, the prevalence of *undiscovered* vulnerabilities across the software ecosystem would still present additional risk. A proactive approach that focuses on eliminating entire classes of vulnerabilities reduces the potential attack surface and results in more reliable code, less downtime, and more predictable systems. Ultimately, this approach enables the United States to foster economic growth, accelerate technical innovation, and protect national security. Leaving these risks unmitigated comes with a costly price tag and may allow America's adversaries to attempt to take advantage of the circumstances.

To further address these dynamics, President Biden signed the National Cybersecurity Strategy (Strategy) in March 2023, setting forth an affirmative vision for cyberspace and imagining a new approach to solving these long-standing, difficult problems. The Strategy calls for two fundamental shifts in how the United States allocates roles, responsibilities, and resources. First, the Strategy calls for rebalancing the responsibility to defend cyberspace to those most capable and best positioned to reduce risks for all. Second, it notes the need to realign incentives to favor the long-term investments required to make cyberspace more resilient and defensible in the years to come. This Strategy recognizes a once-in-a-generation opportunity to make meaningful progress on these hard cyber problems in this decisive decade.

Since its publication, the Biden-Harris Administration has taken concrete steps toward achieving these two fundamental shifts. The National Cybersecurity Strategy Implementation Plan (NCSIP) puts forth a roadmap of detailed initiatives for the United States Government to drive coordinated action.[i] The National Cyber Workforce and Education Strategy (NCWES), progeny of the Strategy, lays out a plan for employers to grow their cyber workforce and educators to expand access to cyber training.[ii] This report speaks directly to the technical community, including technology manufacturers and academic researchers, illustrating two ways their actions can make significant improvements to the Nation's cybersecurity posture.

★ ★ ★ ★ ★ ★

Programmers writing lines of code do not do so without consequence; the way they do their work is of critical importance to the national interest. This shift from reaction to strategic preparation highlights the enormous influence the technical community can have on the security of a shared digital ecosystem. This report articulates a dual approach: first, in order to reduce memory safety vulnerabilities at scale, creators of software and hardware can secure building blocks of cyberspace. This report focuses on the programming language as a primary building block, and explores hardware architecture and formal methods as complementary approaches to achieve similar outcomes. Second, in order to establish better cybersecurity quality metrics, the research community can address the hard and complex research problem of software measurability. This report explores how such metrics can shift market forces to improve cybersecurity quality across the ecosystem.

This work cannot be accomplished by government action alone. These approaches will be ambitious undertakings that will require persistent, multi-sector focus for the years to come. As these crucial efforts move forward, there are grounds for optimism in the ability to overcome the challenges that lie ahead. There are no "silver bullets" in cybersecurity, but power comes through the alignment of today's resources with tomorrow's aspirations. The future prosperity and security of the digital ecosystem requires determined cooperation with the technical community.

PART II: SECURING THE BUILDING BLOCKS OF CYBERSPACE

To reduce the burden currently placed on end users to protect themselves from cybersecurity threats, efforts must be made to proactively eliminate entire categories of software vulnerabilities. To better understand the prevalence of these categories, software manufactures should consider publishing timely, complete, and consistent Common Vulnerability and Exposures (CVEs) data, including the Common Weakness Enumeration (CWE). Past analysis of CVE data identified memory safety bugs as one of the most pervasive classes of vulnerabilities that has plagued cyber defenders for decades.

Memory safety vulnerabilities are a class of vulnerability affecting how memory can be accessed, written, allocated, or deallocated in unintended ways.[iii] Experts have identified a few programming languages that both lack traits associated with memory safety and also have high proliferation across critical systems, such as C and C++.[iv] Choosing to use memory safe programming languages at the outset, as recommended by the Cybersecurity and Infrastructure Security Agency's (CISA) Open-Source Software Security Roadmap is one example of developing software in a secure-by-design manner.[v]

There are two broad categories of memory safety vulnerabilities: spatial and temporal. Spatial memory safety issues result from memory accesses performed outside of the "correct" bounds established for variables and objects in memory. Temporal memory safety issues arise when memory is accessed outside of time or state, such as accessing object data after the object is freed or when memory accesses are unexpectedly interleaved.[vi] Many of the major cybersecurity vulnerabilities over the past several decades were facilitated by memory safety vulnerabilities, including the Morris Worm of 1988, the Slammer Worm denial-of-service attack in 2003, the Heartbleed vulnerability in 2014, and the BLASTPASS exploit chain of 2023.[vii] For over 35 years, this same class of vulnerability has vexed the digital ecosystem.

The highest leverage method to reduce memory safety vulnerabilities is to secure one of the building blocks of cyberspace: the programming language. Using memory safe programming languages can eliminate most memory safety errors. While in some distinct situations, using a memory safe language may not be feasible – this report examines space systems as a unique edge case and identifies memory safe hardware and formal methods as complementary ways to achieve a similar outcome – in most cases, using a memory safe programming language is the most efficient way to substantially improve software security.

★ ★ ★ ★ ★ ★

Memory Safe Programming Languages

Since many cybersecurity issues start with a line of code, one of the most effective ways to address those issues is by examining the programming language itself. Ensuring that a programming language includes certain properties, such as memory or type safety, means software built upon that foundation automatically inherits the security those features provide.

Cybersecurity solutions should be informed by engineering best practices, and technology manufacturers building software can tackle this issue by consistently using secure building blocks; specifically, adopting memory safe programming languages. There is strong evidence that now is the time to make these changes. First, technical solutions already exist; there are dozens of memory safe programming languages that can – and should – be used. Technology manufacturers are already able to design and build new products in memory safe programming languages from day one. Second, the transition to memory safe programming languages has a demonstrably positive effect on cybersecurity. Industry analysis has shown in some cases, that despite rigorous code reviews as well as other preventive and detective controls, up to 70 percent of security vulnerabilities in memory unsafe languages patched and assigned a CVE designation are due to memory safety issues.[viii] When large code bases are migrated to a memory safe language, evidence shows that memory safety vulnerabilities are nearly eliminated.[ix]

For new products, choosing to build in a memory safe programming language is an early architecture decision that can deliver significant security benefits. Even for existing codebases, where a complete rewrite of code is more challenging, there are still paths toward adopting memory safe programming languages by taking a hybrid approach. For example, software developers can identify the critical functions or libraries based on risk criteria and prioritize efforts to rewrite those first.[x]

Building new products and migrating high-impact legacy code to memory safe programming languages can significantly reduce the prevalence of memory safety vulnerabilities throughout the digital ecosystem.[xi] To be sure, there are no one-size-fits-all solutions in cybersecurity, and using a memory safe programming language cannot eliminate every cybersecurity risk. However, it is a substantial, additional step technology manufacturers can take toward the elimination of broad categories of software vulnerabilities. A recent report authored by CISA, the NSA, the FBI, and international cybersecurity agencies entitled *The Case for Memory Safe Roadmaps*, provides guidance for manufacturers with steps to implement changes to eliminate memory safety vulnerabilities from their products.[xii]

Memory Safe Hardware

In April of 1970, an on-board explosion derailed Apollo 13's mission to the Moon. Two days into the astronauts' voyage, an exposed wire ignited a fire and caused one of the ship's two oxygen tanks to burst.[xiii] The astronauts' only hope for survival was for the rocket scientists to react ingeniously – and fast. Employing the laws of physics and the rules of mathematics, the aerospace engineers worked quickly and calculated an engine burn by the lunar module to successfully get Apollo 13 on a trajectory back to Earth.[xiv]

Over fifty years later, aerospace engineers and policymakers alike have not left the future of space safety to fate. Thanks in large part to technological advancements in modern computing and software engineering, digital automation has minimized the risk of human error, shifting the burden away from the astronauts in orbit – and the rocket scientists in the command center – and ensuring the spacecraft is safer by design, and in-turn, safer for its crew. In the case of Apollo 13 the near disaster was inadvertently caused by the laws of physics, but today there are adversaries actively trying to sabotage space systems.[xv] Now, as cyberspace continues to be introduced to outer space, the spacecraft must also be *secure by design*. A catastrophe should not be the catalyst for action.

The space ecosystem is not immune to memory safety vulnerabilities, however there are several constraints in space systems with regards to language use. First, the language must allow the code to be close to the kernel so that it can tightly interact with both software and hardware; second, the language must support determinism so the timing of the outputs are consistent; and third, the language must not have – or be able to override – the "garbage collector," a function that automatically reclaims memory allocated by the computer program that is no longer in use.[xvi] These requirements help ensure the reliable and predictable outcomes necessary for space systems.

According to experts, both memory safe and memory unsafe programming languages meet these requirements. At this time, the most widely used languages that meet all three properties are C and C++, which are not memory safe programming languages. Rust, one example of a memory safe programming language, has the three requisite properties above, but has not yet been proven in space systems. Further progress on development toolchains, workforce education, and fielded case studies are needed to demonstrate the viability of memory safe languages in these use cases. In the interim, there are other ways to achieve memory safe outcomes at scale by using secure building blocks. Therefore, to reduce memory safety vulnerabilities in space or other embedded systems that face similar constraints, a complementary approach to implement memory safety through hardware can be explored.

The chip, in particular, is an important hardware building block to consider. There are several promising efforts currently underway to support memory protections through hardware. For example, a group of manufacturers have developed a new memory-tagging extension (MTE) to cross-check the validity of pointers to memory locations before using them. If they are invalid, the CPU produces an error.[xvii] This technique is an effective method to detect memory safety bugs, but this approach should not be considered a comprehensive solution to prevent all memory safety exploits.[xviii] Another example of a hardware method is the Capability Hardware Enhanced RISC Instructions (CHERI).[xix] This architecture changes how software accesses memory, with the aim of removing vulnerabilities present in historically memory unsafe languages.[xx]

★ ★ ★ ★ ★ ★

Formal Methods

Even if engineers build with memory safe programming languages and memory safe chips, one must think about the vulnerabilities that will persist even after technology manufacturers take steps to eliminate the most prevalent classes. Given the complexities of code, testing is a necessary but insufficient step in the development process to fully reduce vulnerabilities at scale. If correctness is defined as the ability of a piece of software to meet a specific security requirement, then it is possible to demonstrate correctness using mathematical techniques called *formal methods*. These techniques, often used to prove a range of software outcomes, can also be used in a cybersecurity context and are viable even in complex environments like space. While formal methods have been studied for decades, their deployment remains limited; further innovation in approaches to make formal methods widely accessible is vital to accelerate broad adoption. Doing so enables formal methods to serve as another powerful tool to give software developers greater assurance that entire classes of vulnerabilities, even beyond memory safety bugs, are absent.

While there are several types of formal methods that span a range of techniques and stages in the software development process, this report highlights a few specific examples. *Sound static analysis* examines the software for specific properties without executing the code.[xxi] This method is effective because it can be used across many representations of software, including the source code, architecture, requirements, and executables. *Model checkers* can answer questions about a number of higher-level properties.[xxii] These algorithms can be used during production; however, they are limited in their scaled use due to their computational complexity. *Assertion-based testing* is a formal statement of properties carried in the code that may be used to cross-check the code during testing or production.[xxiii] These generated proofs allow for faults to be detected much earlier and closer to the erroneous code, rather than tracing back from externally visible systems failures.

There are two ways software engineers can use these techniques across software and hardware. First, formal methods can be incorporated directly into the developer toolchain. As the programmer builds, tests, and deploys software, the compiler can automate these mathematical proofs and verify that a security condition is met.[xxiv] Additionally, the developer can use formally verified core components in their software supply chain.[xxv] By choosing provably secure software libraries, developers can ensure the components they are using are less likely to contain vulnerabilities.

Formal methods can be incorporated throughout the development process to reduce the prevalence of multiple categories of vulnerabilities. Some emerging technologies are also well-suited to this technique.[xxvi] As questions arise about the safety or trustworthiness of a new software product, formal methods can accelerate market adoption in ways that traditional software testing methods cannot. They allow for proving the presence of an *affirmative* requirement, rather than testing for the absence of a *negative* condition.[xxvii]

While memory safe hardware and formal methods can be excellent complementary approaches to mitigating undiscovered vulnerabilities, one of the most impactful actions software and hardware manufacturers can take is adopting memory safe programming languages. They offer a way to eliminate, not just mitigate, entire bug classes. This is a remarkable opportunity for the technical community to improve the cybersecurity of the entire digital ecosystem.

PART III: ADDRESSING THE SOFTWARE MEASURABILITY PROBLEM

To make progress toward securing the digital ecosystem, it is necessary to realign incentives to favor long-term investments. For this realignment to generate ecosystem-wide behavior change, it is critical to develop empirical metrics that measure the *cybersecurity quality* of software. This will help inform both producer and consumer decision-making, as well as public policymaking efforts. Ongoing work to improve how software quality and security are understood, including coordinated vulnerability disclosure, response programs, and timely CVE records, informs essential decision making throughout the ecosystem. Nevertheless, more progress is required to develop empirical metrics to effectively measure code.

Software measurability is one of the hardest open research problems to address; in fact, cybersecurity experts have grappled with this problem for decades. The problem requires not only refining existing metrics or tools, but also the pioneering of a new frontier in software engineering and cybersecurity research. By advancing capabilities to measure and evaluate software security, more vulnerabilities can be anticipated and mitigated before software is released. The metrics developed from these measurements will also inform the decision-making of a broad range of stakeholders, further improving the security of the digital ecosystem and incentivizing long-term investments in secure software development.

Challenges with Software Measurability

In 2016, the software quality group at the United States National Institute of Standards and Technology (NIST) convened a workshop to explore enhancements in software measures and metrics, aiming to dramatically reduce software vulnerabilities.[xxviii] The conclusions from this workshop underscored the multifaceted nature of the software measurability problem and identified three core challenges: software metrology, software behavior, and software analysis.

First, the inherent challenge of software metrology – the science of software measurement – stems from the fact that software is not just a technical construct, but also a form of human expression. Unlike physical engineering products, most software lacks a uniform structure or composition. This heterogeneity in design and architecture renders the definition of cybersecurity quality highly subjective and context-dependent, complicating the establishment of universal metrics.[xxix]

Second, the behavior of software under various conditions may not be entirely deterministic.[xxx] While certain inputs and processes might predictably lead to specific outputs, the overall behavior of complex software systems often eludes such predictability. Moreover, software does not necessarily conform neatly to probabilistic distributions, making it difficult to apply statistical models or predictions commonly used in other scientific disciplines. This unpredictability hinders the capacity to reliably and consistently measure software performance and security.

Third, the difficulty of analyzing software compounds the problem.[xxxi] Analyzing software to evaluate its cybersecurity quality is limited by what can be quantified. Traditional methods, like counting known vulnerabilities, are insufficient and do not necessarily provide insight into future threats or attack vectors.[xxxii] The dynamic and evolving nature of both software development and cyber threats means that what is measured today may not be relevant tomorrow, making the development of forward-looking, predictive measures an arduous task.

These challenges are substantial, but the research community should revisit this hard problem. The recently published Federal Cybersecurity Research and Development Strategic Plan also highlights software measurability as a research priority for the Biden-Harris Administration.[xxxiii] Increasing the research community's attention on this endeavor will lay the groundwork for the identification and development of measures of software cybersecurity quality. This could enable the creation of valuable metrics that close a significant information gap. While these metrics would be useful for the software developer looking to secure their code, they also have far-reaching implications for the entire cybersecurity ecosystem.

Applications of Cybersecurity Quality Metrics

In December 2021, the discovery of a vulnerability within Log4j, a ubiquitous open-source Java logging library, highlighted a critical weakness through which malicious actors could compromise computer systems across the world.[xxxiv] Governments, multinational corporations, and critical infrastructure operators scrambled to assess the extent of their exposure. This vulnerability highlighted the critical need to help ensure the security of the open-source ecosystem, which fosters tremendous innovation worldwide. Known as "Log4Shell," it illuminated an unfortunate truth in cybersecurity: preemptively measuring the cybersecurity quality of software remains a Herculean task.

The open-source software ecosystem provides a unique backdrop for the examination of software measurement. Open-source software is ubiquitous in hardware and software across nearly every economic sector, which presents a variety of security challenges. The accessible and transparent nature of open-source software also makes it an excellent environment for the application of software measurement and the resulting development of cybersecurity quality metrics.

If developed, robust software measurements could improve one's ability to evaluate cybersecurity quality. By quantifying what was once qualitative, the precision and objectivity of software assessment is improved. This allows for a more nuanced understanding of where and how cybersecurity can be enhanced.

With an effective measurement of quality, the next advancement lies in the continuous monitoring of this metric. Instead of one-time cybersecurity assessments, software can be dynamically evaluated. This advancement would help keep pace with an environment where threats constantly emerge and evolve, and software itself is in a state of perpetual development and refinement.

metrics into a visual narrative of software security. The power of visualization in conveying complex data is well-established, and in the context of cybersecurity, proves invaluable. With a quality graph, trends and patterns that might have remained obscured in tables of data would become more comprehensible.

Finally, with the visualization of quality trends and with an appropriate mathematical function to fit the data, additional metrics can be derived. One potential derivative – response rate – shows the rate of change of the cybersecurity quality metric. This type of derived metric would offer deeper insights into the software's security profile. For instance, a rapid response rate to emerging vulnerabilities would indicate a proactive and trustworthy software vendor.

The application of cybersecurity quality metrics marks a shift in how software security is approached and understood. It is a journey from subjective assessment to objective precision, static snapshots to dynamic trends, and diffuse data to actionable insights. When applied to the opensource software ecosystem it is easy to imagine the resulting impact; like using these metrics to identify an open-source library with poor cybersecurity quality and deciding to use a more secure component instead.

Shifting Market Forces to Improve Cybersecurity Quality

Reframing the discussion on cybersecurity from a reactive to a proactive approach enables a shift in focus from the front-line defenders to the wide range of individuals that have an important part to play in securing the digital ecosystem. For far too long, primary responsibility for the cybersecurity of an organization has rested with the Chief Information Security Officer (CISO) of the company *using* software. They cannot be the only stakeholder accountable for cybersecurity outcomes; it is also critical, for example, that the Chief Information Officer (CIO) who is *buying* software, and the Chief Technology Officer (CTO) of manufacturers *building* software share this responsibility. A cybersecurity quality metric could improve collaborative decision-making across all parties.

In navigating the complex research landscape of software measurability, there are three fundamental dimensions to the risk software poses to the cybersecurity of an organization: the developer process, the software analysis and testing, and the execution environment.[xxxv] When looking at the development of a quality metric, the first two dimensions are fundamentally reflective of the software itself and are where architectural and design choices play the most significant role in improving security. They are also the dimensions that will remain static across all deployments of a version of software. Understanding these three dimensions can inform how cybersecurity quality metrics can be used across the ecosystem.

The CTOs of software manufacturers and the CIOs of software users are best leveraged to make decisions about the intrinsic quality of the software, and are therefore likely most interested in the first two dimensions of cybersecurity risk. In the first dimension, the software development process, the caliber of the development team plays a crucial role. Teams that are well-trained and experienced, armed with clear requirements and a history of creating robust software with minimal vulnerabilities, foster a higher level of confidence in the software they produce.[xxxvi] The competence and track record of the development team serve as hallmarks of reliability, suggesting that software crafted under their expertise is more likely to be secure and less prone to vulnerabilities. A CTO might make decisions about how to hire for or structure internal development teams to improve the cybersecurity quality metrics associated with products developed by the organization, and a CIO may make procurement decisions based on their trust in a vendor's development practices.

The second dimension that both CTOs and CIOs could consider is the analysis of the software product. With better metrics, CTOs could use a range of rigorous analysis methods – such as code reviews, acceptance tests, and formal methods – to assure that vulnerabilities in a piece of software will be rare. A CIO might base a purchasing decision in large part on how well a software product scores on quality metrics, confident that its adoption would pose less risk to the organization.

The CISO of an organization is primarily focused on the security of an organization's information and technology systems. While this individual would be interested in all three dimensions of software cybersecurity risk, they have less direct control over the software being used in their environments. As such, CISOs would likely be most interested in the third dimension: a resilient execution environment. By running the software in a controlled, restricted environment such as a container with limited system privileges, or using control flow integrity to monitor a program at runtime to catch deviations from normal behavior, the potential damage from exploited vulnerabilities can be substantially contained.[xxxvii] This method does not eliminate vulnerabilities but rather mitigates their impact, serving as a potential safety net in case of exploitation.[xxxviii] Since the CISO does not always get to decide what technology and security capabilities they work with, this dimension could influence a CISO's prioritization of risk mitigation actions for a piece of software with problematic cybersecurity quality.

While the Biden-Harris Administration has taken significant steps to motivate the market to patch known vulnerabilities, including Executive Order 14028, users still face cybersecurity risk from their software due to a lack of information that can help reduce future vulnerabilities – by stopping them before they occur, by finding them before they are exploited, or by reducing their impact.[xxxix] Software manufacturers are not sufficiently incentivized to devote appropriate resources to secure development practices, and their customers do not demand higher quality software because they do not know how to measure it. Operationally, organizations do not take sufficient steps to mitigate the risk of low-quality software because they are not aware they are carrying that risk at all and often cannot afford the mitigations even if they are aware.

Better cybersecurity quality metrics change the equation because they will enable data-informed decision-making across the supply chain. While the technical executives, like the CTO, CIO, and CISO, play a defining role in executing this vision, cybersecurity quality must also be seen as a business imperative for which the CEO and the board of directors are ultimately accountable. Addressing the software measurability problem would fully realize this metric's utility, closing a vital information gap and incentivizing long-term investments in software security. This would allow all ecosystem stakeholders to see their return on investment or clearly understand the risk of a lower quality product.

PART IV: CONCLUSION

The challenge of eliminating entire classes of software vulnerabilities is an urgent and complex problem. Looking forward, new approaches must be taken to mitigate this risk. Doing so will allow the United States to continue its progress toward President Biden's affirmative vision for a secure and resilient cyberspace.

The technical community is critical to this progress. Through the adoption of memory safe programming languages, creators of software and hardware can better secure the building blocks of cyberspace and proactively eliminate entire classes of bugs. By rallying around the hard and complex problem of software measurability, the research community can develop better cybersecurity quality metrics to incentivize better decision-making by consumers, manufacturers, and policymakers across the ecosystem. These efforts will be bold, long-term endeavors that require sustained focus and prioritization. Now is the time to begin this work.

The road toward this vision requires a recognition that the Nation is at its best when Americans work together. It is a path that requires the convergence of government initiative, private sector innovation, and groundbreaking academic research. Working together to proactively eliminate software vulnerabilities alleviates the burden from those least equipped to bear it, and empowers front-line cyber defenders to look forward. Defining high-quality cybersecurity realigns incentives and provides confidence in what cyberspace can be. Together, these collaborative forces will continue to propel us toward a future where cyberspace is secure, resilient, and trustworthy, embodying fundamental American values and remaining unbent to the animosity of our adversaries. As President Biden frequently remarks, "We are the United States of America and there is nothing, nothing beyond our capacity if we do it together."

PART VI: ENDNOTES

[i] *The National Cybersecurity Strategy Implementation Plan*, The White House, July 2023 *available at* NCSIP.WH.GOV.

[ii] *The National Cyber Workforce and Education Strategy*, The White House, July 2023, *available at* NCSWES.WH.GOV.

[iii] *The Case for Memory Safety Roadmaps,* Department of Homeland Security, Cybersecurity Infrastructure and Security Agency, December 2023, *available* here [hereinafter Roadmaps].

[iv] *See id*. *See also* Introduction to Memory Unsafety for VPs of Engineering · Alex Gaynor, August 2019, *available* here.

[v] *See CISA Open Source Software Security Roadmap*, Department of Homeland Security, Cybersecurity Infrastructure and Security Agency September 2023, *available* here.

[vi] *See Report to the CISA Director*, Department of Homeland Security, Cybersecurity Infrastructure and Security Agency, December 2023, at page 2, *available* here.

[vii] *See generally* Morris worm - Wikipedia. *See also* National Vulnerability Database, Department of Commerce, National Institute of Standards and Technology, *available* here. *See also BLASTPASS: NSO Group iPhone Zero-Click, Zero-Day Exploit Captured in the Wild* , University of Toronto, Munk School of Global Affairs and Public Policy, September 2023, *available* here.

[viii] *See Microsoft MSRC Blog We Need a Safer Systems Programming Language*, July 2019, *available* here.

[ix] *See Memory Safe Languages in Android 13*, Google Security Blog, December 2022, *available* here.

[x] *See generally About Prossimo, available here*. A risk criteria framework to prioritize code rewrites can be defined as software that is "1. very widely used, 2. on a network boundary, 3. performing a critical function, [and] 4. written in languages that are not memory safe." *Available* here.

[xi] *See generally Google Project Zero: The More You Know, The More You Know You Don't Know*, April 2022, *available* here.

[xii] *See* Roadmaps *supra* note iii.

[xiii] *See* Corum, Jonathan, "Apollo 13 As They Shot It." *New York Times*, 11 Apr. 2020, *available* here.

[xiv] *See* Chang, Kenneth, "Apollo 13's Astronauts Survived Disaster 50 Years Ago. Could It Happen Again?" *New York Times*, 13 Apr. 2020, *available* here.

[xv] Press Release, Foreign, Commonwealth, and Development Office of the United Kingdom, 10 May 2022, *available* here.

[xvi] *Readout of Acting National Cyber Director Kemba Walden's Travel to California*, The White house, May 2023, *available* here.

[xvii] *See generally Adopting the Arm Memory Tagging Extension in Android*, Google Security Blog, August 2019, *available* here.

[xviii] *See generally Arm Memory Tagging Extension*, Android Security, *available* here.

[xix] *See Capability Hardware Enhanced RISC Instructions (CHERI)*, University of Cambridge, Department of Computer Science and Technology, *available* here.

[xx] *See generally The New CHERI-enabled Morello Boards*, SRI, March 2022, *available* here.

[xxi] *See* Paul E. Black, et al., Department of Commerce, National Institute of Standards and Technology, *Dramatically Reducing Software Vulnerabilities*, NIST IR 8151, November 2016. *available* here. [Hereinafter NIST IR 8151]. [xxii] *Id*.

[xxiii] *Id*.

[xxiv] *Id*.

[xxv] *See generally*, Microsoft Research – Project Everest, *available* here.

[xxvi] *See generally*, Autoformalization with Large Language Models, Cornell University, May 2022, *available* here, *see also* A New Era in Software Security: Towards Self-Healing Software via Large Language Models and Formal Verification, Cornell University, May 2022, *available* here.

[xxvii] *See Clover: Closed-Loop Verifiable Code Generation*, Cornell University, October 2023, *available* here.

[xxviii] Paul E. Black and Elizabeth Fong, Department of Commerce, National Institute of Standards and Technology, NIST Special Publication 500-320, *Report of the Workshop on Software Measures and Metrics to Reduce Security Vulnerabilities*, November 2016. *Available* here. [hereinafter NIST Special Publication].

[xxix] *Id*. at 19.

[xxx] *Id.*
[xxxi] NIST Special Publication *supra* note xxvii at 24.
[xxxii] *Id.* at 59.
[xxxiii] *See Federal Cybersecurity Research and Development Strategic Plan*, National Science and Technology Council, The White House, December 2023 *available here*.
[xxxiv] *See* Bloomberg "Inside the Race to Fix a Potentially Disastrous Software Flaw", December 2021. *Available here*.
[xxxv] NISTIR 8151 *supra* note xxii at 38.
[xxxvi] *Id.*
[xxxvii] *See* Ruan de Clercq and Ingrid Verbauwhede. "A Survey of Hardware-based Control Flow Integrity (CFI)" *available here*.
[xxxviii] NISTIR 8151 *supra* note xxvii at 38.
[xxxix] *Id.* at ii.

THE WHITE HOUSE

WASHINGTON