Proposal: Leveraging Artificial Intelligence to Enhance Cybersecurity by Translating Memory-Unsafe Languages into Rust for Government and Critical Infrastructure

Executive Summary

Software vulnerabilities arising from memory-unsafe programming languages (such as C and C++) pose a significant national security risk—studies attribute up to 70% of all security bugs to memory safety flaws. These flaws have enabled some of the most consequential cyberattacks in recent history. In order to address this challenge, it is proposed that the federal government undertake an AI-driven initiative to automatically translate legacy governmental software from memory-unsafe languages into Rust, a modern memory-safe language, by the year 2030. By harnessing advanced AI (including large language models) to perform code translation at scale, it is possible to eliminate an entire class of vulnerabilities and substantially reduce cybersecurity risks. This undertaking would bolster national security, reduce long-term software maintenance costs, and improve the reliability of critical systems while aligning with federal cybersecurity policies that call for more secure "memory safe" software development. The following sections outline the rationale, approach, phased implementation plan, security considerations, expected benefits, challenges, and overarching recommendations for this transformative effort.

Background and Rationale

Memory management bugs (for example, buffer overflows or use-after-free errors) comprise the most prevalent category of software vulnerabilities. Historically, the response has been to rely on patching and bug-hunting, but national cybersecurity leadership, including the White House Office of the National Cyber Director (ONCD), has emphasized the need for proactive solutions that address the root cause of memory safety flaws. By preventing memory errors, attackers lose access to the exploits they frequently rely upon.

Memory-safe languages like Rust and Go offer inherent protections that block entire classes of memory-related vulnerabilities. However, substantial amounts of legacy C/C++ code remain deeply integrated across the government's critical infrastructure and defense systems, making a complete manual rewrite impractical. Recent advances in AI—particularly large language models (LLMs)—offer the potential to facilitate automated code translation from C/C++ to Rust at scale.

In 2024, the Defense Advanced Research Projects Agency (DARPA) launched the "Translating All C to Rust" (TRACTOR) program, reflecting the feasibility of this approach. Building on those foundational efforts, it is recommended that the government capitalize on next-generation AI to modernize its extensive legacy software, thereby removing memory-corruption vulnerabilities at their foundation. This approach aligns directly with the ONCD's call to reduce the cyberattack surface through proactive measures rather than reactive patches.

Technological Approach

The core of this initiative relies on Large Language Models (LLMs) trained on extensive corpora of software code. These models can already perform language-to-language translation of code (for example, from C to Rust). Although the current accuracy is not perfect, specialized AI systems fine-tuned for memory-safe code generation can improve results dramatically. It is suggested that the government develop AI "agents" that orchestrate several tools in tandem: the LLM to perform initial translation, static analysis tools to check the translated code, and symbolic execution or other verification tools to ensure functionality and security requirements match the original program.

As the capability of AI models improves, these agents could address increasingly intricate code scenarios, including low-level pointer arithmetic and inline assembly, by applying safety wrappers or specialized modules. Government-sponsored competitions—similar to DARPA's public contests—could further drive advancements and engagement, optimizing and refining AI translators over time. By the late 2020s, AI translation is projected to require considerably less human intervention, enabling high-fidelity conversion of legacy systems into memory-safe Rust implementations.

Implementation Plan

1. **Phase 1 (2024–2025) – Research and Preparation**
    - Expand and fund programs such as DARPA's TRACTOR (initiated in 2024) to develop robust C-to-Rust translation tools.
    - Direct all federal agencies, in collaboration with critical software vendors, to inventory their legacy code and publish a **Memory Safety Roadmap** by the end of 2025.
    - Establish standards and best practices for AI-translated code, including Rust coding standards and testing protocols.
    - Conduct pilot projects using non-mission-critical systems to evaluate and refine initial AI translation tools.
2. **Phase 2 (2026–2028) – Pilot Conversions and Integration**
    - Prioritize translation of high-risk software components (e.g., network-facing modules, memory-intensive services, cryptographic libraries) using matured AI translation tools.
    - Integrate translated Rust modules into larger systems incrementally, verifying interoperability with any remaining C/C++ code.
    - Enhance AI tools with continuous feedback loops, retraining the models upon discovery of any issues or inaccuracies.
    - Ensure that all new development defaults to memory-safe languages to prevent further accumulation of memory-unsafe technical debt.
3. **Phase 3 (2029–2030) – Widespread Adoption and Enforcement**
    - Require that, by 2029, all new government software be developed in memory-safe languages.

- Complete translation of remaining critical legacy software, isolating or sandboxing any components that cannot be fully converted due to technical constraints.
- Enforce compliance through audits and oversight measures; assist agencies facing obstacles to ensure alignment with the 2030 objective of nearly eliminating memory-corruption exploits from federal and critical infrastructure systems.

Progress during each phase is to be measured against clear milestones (e.g., percentage of code converted, vulnerabilities eliminated, performance benchmarks) to ensure accountability and guide timely adjustments.

Security and Compliance

This proposal directly supports national cybersecurity policies and secure-by-design principles. The 2024 ONCD report "Back to the Building Blocks" urged migration to memory-safe languages to remove entire categories of vulnerabilities. Further, the Cybersecurity and Infrastructure Security Agency (CISA) has classified reliance on memory-unsafe code in critical software as an unacceptable risk.

The initiative would build on existing directives that require vendors and federal agencies to publish memory safety roadmaps and implement them as a condition of procurement. All AI-translated code would undergo comprehensive reviews, including static and dynamic security testing. Source code, particularly sensitive or classified materials, would be processed in secure environments to maintain confidentiality.

By systematically converting legacy software to Rust, government systems would reflect the "Secure by Design" philosophy. Oversight by a governance board (including representatives from ONCD, CISA, the Department of Defense, and other agencies) would verify compliance with relevant federal security standards (for instance, NIST recommendations, FIPS validations, and SSDF guidelines). This structure aims to ensure that the entire migration process aligns with best practices and strengthens national security.

Expected Benefits

1. **Significant Cybersecurity Advancement**
   By focusing on eliminating memory-corruption bugs, adversaries would lose one of the most prevalent attack vectors. Critical infrastructure and defense systems would become substantially more resilient to both criminal and nation-state cyber threats.
2. **Cost Savings and Efficiency**
   Preempting vulnerabilities is far more economical than reacting to them. With fewer security breaches to patch and mitigate, software maintenance costs would be reduced. Leveraging AI to automate code translation is also far more efficient than manual rewrites, thereby lowering modernization expenses.
3. **Improved Reliability and Performance**
   Rust's compile-time checks and memory-safety guarantees reduce runtime crashes, memory leaks, and other disruptions. Systems converted to Rust often match or exceed

C/C++ performance, particularly for multi-threaded applications, resulting in improved system stability for both government personnel and the public.

4. **Strategic Technology Leadership**
   By orchestrating a large-scale shift to memory-safe software with AI assistance, the federal government would set an industry-leading example of secure development. This effort would encourage broader adoption of safe languages and advanced AI tools, elevating national competitiveness and paving the way for continued innovation.

Challenges and Considerations

1. **Accuracy of AI Code Translation**
   Although AI translators are advancing rapidly, inaccuracies remain possible. Rigorous testing, verification, and, initially, human oversight are recommended to ensure correctness. Continuous improvement of AI models through real-world feedback will reduce risk over time.
2. **Scale and Complexity of Legacy Systems**
   The sheer volume and diversity of legacy code introduce both technical and logistical hurdles. It is recommended that the government adopt a phased approach, addressing the most critical systems first and employing wrappers or partial refactoring if certain low-level code cannot be fully automated.
3. **Integration and Compatibility**
   Translated Rust modules must seamlessly interoperate with existing components that remain in C/C++. Rust's Foreign Function Interface (FFI) will be key for phased integration, with careful testing to ensure minimal disruption to mission-critical services.
4. **Workforce and Training**
   A shift to Rust and AI-based workflows constitutes a cultural and skill-based transition. A comprehensive training strategy—including bootcamps, certifications, and incentives—would be required to expand Rust expertise across federal agencies and contractor teams.
5. **Policy and Operational Constraints**
   Safety-critical systems (such as those in aerospace or defense) may require lengthy re-certification processes when introducing translated Rust code. Additionally, intellectual property rights and licensing agreements must be reviewed to ensure that automated translation does not violate third-party terms. Thorough project management and inter-agency coordination are crucial to address these constraints.

Conclusion and Recommendations

Transitioning government and critical infrastructure software to memory-safe languages through AI-driven code translation constitutes a bold yet feasible strategy for eliminating a predominant source of cybersecurity vulnerabilities. By 2030, this approach can realize a future where memory-corruption exploits are largely eradicated from federal systems, significantly strengthening the nation's cyber posture. The following key recommendations aim to guide this effort:

1. **Institute Formal Memory-Safe Coding Policies**
   Update procurement and development regulations to require memory-safe languages for all new software, discouraging C/C++ development in critical areas. Ensure that each agency maintains a detailed Memory Safety Roadmap, with clear milestones and enforced oversight.
2. **Fund and Expand AI Translation Research**
   Invest in large-scale R&D, building upon DARPA's TRACTOR program and similar efforts across the National Science Foundation, Department of Defense, Department of Homeland Security, and other agencies. Create public-private partnerships and competitions to accelerate progress and refine AI-enabled translation tools.
3. **Develop Workforce Expertise**
   Provide ample training programs and resources to ensure that government developers, contractors, and related personnel can effectively use Rust and AI-assisted development tools. Establish a Rust Center of Excellence within a federal agency or as a collaborative, cross-agency resource.
4. **Monitor Compliance and Measure Impact**
   Form a governing body (led by the Office of the National Cyber Director in collaboration with CISA and the Office of Management and Budget) to track implementation, audit progress, and measure metrics such as the reduction in vulnerabilities, cost savings, and performance improvements. Include memory-safety migration status as part of existing federal security reviews.
5. **Promote Wider Adoption Beyond Federal Systems**
   Encourage or require vendors and private-sector partners that provide critical software to adopt memory-safe practices. Ensure that commercial off-the-shelf and open-source software used by the government also transition away from memory-unsafe languages, thereby securing the broader software supply chain.

With sustained leadership, investment, and collaboration, the government can leverage AI to systematically convert its foundational software to memory-safe Rust implementations, effectively neutralizing an entire class of cyber threats by 2030. This decisive action would position the United States at the forefront of secure software engineering, yielding lasting benefits for national security, economic stability, and public trust in government services.