Green Hills Software Submission

Position Paper on Standards and Guidelines to Enhance Software Supply Chain Security

Submitted to:

Submitted By:

30 W. Sola Street

Santa Barbara, Ca. 93101

Robert O'Dowd, Business Strategist

1 (805) 965-6044

05/26/2021

Green Hills Software is a provider of key software technologies for US military, intelligence community, and commercial aviation for certified safety and security of mission critical software. Green Hills Software provides empirically vulnerability free software and an auditable codebase to facilitate rigorous examination of source code. These activities provide assurances of reliability at the highest levels of certification for aviation, automotive, industrial control, and information technology.

Green Hills Software's INTEGRITY separation kernel technology is the first and only software technology to be certified to Evaluated Assurance Level (EAL) 6+ / High Robustness, the highest security level ever achieved for any software product under the ISO 15408 (Common Criteria) standard. High Robustness is defined by the DoD as appropriate for protection of high value resources against the most sophisticated attackers[1]. Green Hills Software has extensive experience developing high assurance products satisfying both the Common Criteria and other high assurance certification standards, including DO-178B Level A, the highest safety level for commercial avionics. Other software developed to high assurance standards by Green Hills Software includes file systems, internet protocol networking stacks, device drivers, access cross domain solutions, network security protocols, and cryptographic functions.

1. Criteria for designating "critical software"

The FAA DO-178B[2] Design Assurance Level conducts a safety assessment and hazard analysis for every software component in a system and assigns levels of criticality based on the effects on aircraft viability. The framework divides impact into levels ranging from Catastrophic to No Effect and outlines objectives and tolerable failure rates for each sub-system. This framework can be adapted for any software architecture where security and reliability must be evaluated. Systems with multiple software components must be evaluated based on the scope of impact for each individual component in the event of failure. Vulnerabilities in critical software should be viewed as defects and remediated as such in the event of discovery in a widely deployed life critical application.

Critical software must be categorized based on impact level if the component in question fails either in an accident or hostile attack scenario. For projects of domestic security concern, US military and intelligence community cyber warfare teams should perform threat analysis and testing to evaluate the system's resilience against high threat capability actors. These software assessments should be conducted on all critical aspects of American infrastructure and should include, but is not limited to, sectors such as power and water, transportation, justice, and health and human services. Ultimate decision-making authority on the evaluation and importance of these systems cannot be held by the vendor themselves or the system may be abused by commercial interests over security concerns.

2. Initial list of secure software development lifecycle standards, best practices, and other guidelines acceptable for the development of software for purchase by the federal government.

It is readily accepted that retrofitting security for an existing product line is economically infeasible above the Common Criteria Certification level EAL4[3]. Secure software must be developed with security principles from the beginning. Software products initially designed for non-security focused commercial applications are built by developers whose design principles are not conducive to proving security

[1] Information Assurance Technical Framework, National Security Agency, 2002.
[2] RTCA/DO-178B "Software Considerations in Airborne Systems and Equipment Certification"
[3] Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components, 2017

assurances. These programs often result in poorly documented “spaghetti code” which is impossible to validate and maintain for a security critical environment.

Newly developed software must undergo continuous daily testing against all deployed hardware and software implementations and pass validation tests to ensure continuous compatibility of all update changes to avoid legacy hardware complications. This best practice ensures continuity of functionality throughout development and identification of legacy devices vulnerable to 0-day vulnerabilities.

3. Guidelines outlining security measures that shall be applied to the federal government’s use of critical software

Network segmentation between high and low security domains needs to be enforced by high assurance segmentation technology certified to the same or higher standard than the at-risk domain. Many software systems utilize components which are unintentionally codependent due to security vulnerabilities which permit unrestricted lateral movement, privilege escalation, and arbitrary code execution despite defined scope or permissions. Attackers can and will exploit vulnerabilities in the segmentation technology to bypass security measures or discover an unintended security policy errors to move into a higher security domain.

Even with proper employee cyber awareness training and professional implementation, a highly skilled nation state adversary will invest resources into discovering 0-day vulnerabilities against these network segmentation technologies. Therefore, these network segmentation technologies must target development to approach a vulnerability-free state in order to effectively deter capable adversaries. Part of this effort should include formal verification of software systems for mathematical correctness, especially in software applications of national security priorities.

4. Initial minimum requirements for testing software source code

Minimum requirements should be established for software components at each tier of impact within a system. The minimum requirements for any given software program should be defined by the software impact on safety and security, with high impact projects being analyzed with certified tools to maximize debugging effectiveness. Emphasis should be placed on development tools to facilitate early identification of bugs during the coding and pre-release testing phases to minimize the impact of incidents of vulnerabilities discovered post-deployment.[5] For systems identified as “critical software,” high assurance testing methodologies such as covert channel mitigations, worst case execution timing, abstract machine testing and interference analysis are examples of minimum testing needed.

5. Guidelines for software integrity chains and provenance

Deployed systems need to be managed with a device lifecycle management framework to ensure remote systems remain up to date. The update mechanisms need to be cryptographically enforced to only permit authorized updates from the principal developer only. Sophisticated man-in-the-middle attacks can masquerade as legitimate signals to devices and exploit remote updating mechanisms to run malicious software. Properly keyed devices using products analogous to Type 1 encrypters can validate only authorized code to be loaded onto managed systems.

[5] NIST 2002, “The Economic Impacts of Inadequate Infrastructure for Software Testing”