Broadcom and Symantec (A Division of Broadcom) Position Paper on Standards and Guidelines to Enhance Software Supply Chain Security

Issue # 1: *Criteria for designating "critical software."*

Author: Martin Schulman
Title: Security Architect
Email: martin.schulman@broadcom.com

In the context of computer and network security, some may assume "software" refers only to the compiled x86 assembly and other forms (i.e. Java p-code, interpreted high level languages) executing on the main CPU's and optional GPUs running Windows and Linux operating systems. At Broadcom, we know software includes much more than what is typically protected by our Symantec Enterprise Division: from the drivers on our SAS, RAID, FC, and disk controllers to our Application Development & Testing, Identity & Access Management, and Compliance & Data Protection products that protect the nation's financial systems on mainframes to the set-top boxes, cell phones, and other embedded technologies running our ARM-based microcontrollers and other solutions, to our programmable Ethernet Switching & Routing products that connect them all together.

In a systems context, "critical" already refers to any equipment, person, or procedure that is essential to a user, and we see no benefit to changing it. Thus, "critical software" is any set of modifiable instructions that determine how data is stored, transformed, or transmitted and deemed essential to a user's mission, business, safety, or security. Attackers look for all vulnerabilities and often move laterally, so if software running on the main processor of a device is deemed critical, the software on its embedded processors and programmable devices common in today's servers, personal computers, cell phones, network devices, and other infrastructure may also be critical. The same software may be considered critical in one organization but not another, depending how and where it is used.

Which presents a practical problem: if an environment has thousands to millions of networked devices, and each has multiple, independently programmable subsystems, isn't nearly all software critical? If so, will it be possible to apply security measures to all of it within its operational lifetime?

We believe:

- Some software is more critical than others: When considering critical software, we must consider levels of risk. How to measure risk requires deep understanding of a system's architecture, design, and operation, and is beyond the scope of this white paper, though resources like NIST Special Publication 800-30 may help.

- Solutions protecting critical software will require a range of techniques: Cryptographic techniques, software development training, automated development tools, operational attestations, and more will be needed; there is no single magic bullet to protect critical software. Sound network design and proper monitoring tools will be the best current defense for software that would affect business continuity if damaged.

- Two-way sharing is critical: Government and industry must continue to trust each other and share information about attacks, threats, and vulnerabilities.

- More research is needed: Software complexity is already orders of magnitude beyond what we can hope to analyze with automated tools. To prevent the next SolarWinds attack, we should expand funding for academia and incentives for industry research in scalable verification.