CISQ Comments on Promoting Stakeholder Action Against Botnets and Other Automated Threats

In response to:
Department of Commerce and National Telecommunications and Information Administration
[Docket No. 170602536-7536-01]
RIN 0660-XC035
Email: counter_botnet_RFC@ntia.doc.gov

Submitted on July 27, 2017 by:
Consortium for IT Software Quality (CISQ)
Contact: Dr. Bill Curtis, Executive Director, CISQ (director@it-cisq.org)

Objective

This response addresses ***Endpoint Prevention*** by introducing DoC/NTIA to the CISQ/OMG measurement standards for the automated detection, quantification, and remediation of Security and Reliability weaknesses in software-intensive systems that are frequently exploited by botnets and other automated threats. The focus of these comments will be on the internal cybersecurity of endpoint systems and products rather than on the communication infrastructure in which they reside.

Submitting Organization

This response is submitted on behalf of the Consortium for IT Software Quality™ (CISQ™) to the RFC on actions to counter botnets and other automated threats. CISQ is a Special Interest Group managed by the Object Management Group® (OMG®) chartered to develop standards for automating the measurement of size and quality from the source code of software-intensive systems. CISQ was established through a joint action of OMG and the Software Engineering Institute (SEI) at Carnegie Mellon University.

Overview

Critical infrastructure systems are vulnerable to unauthorized penetration when severe source code weaknesses provide easily exploited attack surfaces. Thus, detection, measurement, and remediation of structural quality weaknesses must be a standard practice in all software development, maintenance, acquisition, and related processes to ensure that mission and business-critical services operate on secure, dependable software. Source code-level measurement is critical for accurately assessing the risk of these systems being exploited by botnets and other automated threats.

Background: CISQ Automated Source Code Measurement Standards

During 2015 four CISQ Automated Source Code Measure specifications (Reliability, Security, Performance Efficiency, and Maintainability) were approved as OMG® standards and are available free of charge from the OMG (www.omg.org) and CISQ (www.it-cisq.org) websites. They are the only international IT standards that measure the structural quality of software-intensive systems based on quantifying specific, named weaknesses in the source code. These measures were constructed using conceptual definitions in ISO/IEC 25010 and weaknesses that the initial 24 member companies of CISQ deemed severe enough that they must be removed from the code. The CISQ measures are quantified from the automated analysis of architectural and coding weaknesses in source code using static analysis technology.

Two of the CISQ measures are directly relevant to the threat of botnets and automated attacks - the automated source code measures for Security and Reliability[1,2].

The CISQ Automated Source Code Security Measure is based on the Top 25 Common Weaknesses contained in the Common Weakness Enumeration (CWE) managed by MITRE Corporation on behalf of the cybersecurity community with funding from the Department of Homeland Security (http://cwe.mitre.org). Some of these cybersecurity weaknesses are also known in industry as the 'CWE/SANS Top 25 Most Dangerous Software Errors' and the 'OWASP Top 10'. Of these 25 weaknesses, 22 can be detected through static analysis of the source code. Counting the occurrences of these 22 weaknesses in a software-intensive system constitutes the CISQ Security measure. These weaknesses include such well-known, exploitable flaws as SQL injection, cross-site scripting, and buffer overflows. Vendors of static analysis tools detect most, but not necessarily all these weaknesses, and their coverage increases yearly. To view the specific weaknesses in the Security standard: http://it-cisq.org/wp-content/uploads/2015/08/CISQ-Security-Weakness-Description.pdf

The CISQ Reliability measure is composed from 29 critical violations of good architectural and coding practices that affect the availability, fault tolerance, recoverability, and data integrity of an application. Operational incidents caused by these weaknesses can be exploited by malicious actors and automated threats. To view the 29 weaknesses in the Reliability standard: http://it-cisq.org/wp-content/uploads/2015/08/Reliability-Weakness-Description.pdf

Answers to RFC Questions

1. ***What works:*** In addition to penetration testing, static code analysis of software-intensive systems to detect weaknesses, coupled with disciplined remediation actions to eliminate them are among the most effective cybersecurity practices. The objective is to eliminate, or dramatically reduce attack surfaces in the source code that can be exploited by botnets and other automated threats. Detection is not enough, since time must be planned and protected to systematically eliminate weaknesses at a pace determined by the risk appetite

of agency or business executives. Executives need measures of the number and risk of weaknesses in the software to determine how their critical applications compare to their risk appetite, and to balance the tradeoff between adding new functionality versus eliminating weaknesses. Source code must be measured at the system level since some of the most exploitable weaknesses involve interactions between multiple components across a multi-layer system. Commercial data demonstrates that reductions in incidents and outages correlate with reductions in the number of structural weaknesses within systems.

2. ***Gaps:*** Although they have helped improve the discipline of software development, process-based measures such as CMMI maturity levels have not proven sufficient to ensure cybersecure systems. Unfortunately, ISO/IEC 25023 which defines measures of software security does not specify measures to the level of specific software weaknesses that can be detected and quantified to determine cybersecurity vulnerability and risk. The most accurate measure of the cybersecurity risk of a software-intensive system comes from detecting and measuring weaknesses in code and the rate at which they are removed. One of the most serious problems in software development is the truncation of quality assurance practices to meet schedule and delivery constraints. Gaps in management practices (estimating, planning, tracking, controlling scope, adjusting commitments, enforcing disciplined practices, etc.) that fail to protect the time required to sufficiently evaluate a system against its quality and cybersecurity requirements frequently result in software riddled with weaknesses - some of which compromise cybersecurity. Gaps also exist when cybersecurity practices rely mostly on conventional testing which primarily assesses the correctness of the required functionality. Modern quality assurance and cybersecurity evaluation requires an ensemble of analysis techniques including functional testing, penetration testing, static analysis at the architectural and code levels, dynamic stress testing against a realistic simulation of operational loads, and related practices. Finally, there is a distressing gap in the knowledge of secure coding practices in the software development community. Far too many developers report being self-taught and others were trained in what can best be described as coding camps. Secure coding must be treated as an engineering discipline trained with the same rigor to which any other engineer would be held accountable. IT and product development organizations must assume responsibility to ensure that their developers are skilled in secure coding practices. The Computer Emergency Response Team (CERT) at the SEI, the SANS Institute, and others have developed curricula for secure coding that should be adopted in both government and industry.

3. ***Addressing the problem:*** Reducing the risk of botnets and other automated threats must involve reducing the available attack surfaces in software-intensive systems. The measurement, quality and cybersecurity assurance techniques, and management practices described in the above sections are necessary to reduce these attack surfaces. In addition to these practices, entities acquiring systems must improve their acquisition practices, regardless of whether systems are developed internally or by outside vendors, to ensure the systems they deploy are secure. These acquisition practices involve assessing the maturity and discipline of the development organization's technical and management

practices. System acquisition or maintenance contracts and system requirements should include quantifiable quality and cybersecurity targets. Achievement of these targets must be demonstrated before a software-intensive system is placed in operation. Similarly, Underwriters Lab is developing methods for assessing the quality and cybersecurity of embedded software in commercial network-connectable products. Cybersecurity certification should become a standard practice for systems and products with cyber-risk implications. For software developers who write code for systems or products with cybersecurity implications, certification against secure coding practices should be encouraged if not required.

4. ***Governance and collaboration:*** Governments should update and improve their acquisition practices for software-intensive systems. The state of Texas has just passed a law effective January 1, 2018 requiring that all large state-funded IT systems be measured for progress and quality. Such laws can be enhanced to include requirements for measuring the delivered source code for cybersecurity and dependability risk. Standards organizations such as CISQ, OMG, IEEE, ISACA, and ISO should be enlisted to develop or update standards that address the current and anticipated future automated threats to software-intensive systems. These standards should be developed or enhanced in collaboration with government, FFRDCs, or industry organizations, such as NIST, SEI, MITRE, TIA, UL, etc. These standards must include both process and product based practices and measures. Much of governance and enforcement to ensure cybersecure systems must be achieved through rigorous acquisition, acceptance, and release to production practices. Providing sample procurement language to address removal of software weaknesses and vulnerabilities in software as terms and conditions for acquisition would help organizations in mitigating risks attributable to exploitable software; removing vectors of attack prior to acceptance and use.

5. ***Policy and the role of government:*** Government can be the catalyst for improved cybersecurity practices and systems globally. We saw in the 1990s that the use of the Capability Maturity Model (CMM) in system acquisition by DoD drove global adoption of the standard with a resulting improvement in quality and productivity where the CMM was adopted with the necessary discipline. Similarly, the Federal government can drive cybersecurity practices if it takes a unified stance behind a common set of standards. The CMM experienced global adoption because other Federal agencies followed DoD's lead in applying the CMM. Currently NIST is well-positioned with its Cybersecurity Framework to coordinate a Federal effort to address cybersecurity both at a product and communication ecosystem level. DoC/NTIA could take the lead in developing standards governing the telecommunication infrastructure. However, unless the effort is coordinated across the areas of responsibility among the various Federal agencies, malicious actors will find gaps in the coverage of standards and regulations that can be exploited by automated threats.

6. ***Users:*** As enterprises continue to automate more of their core mission or business functions, executives must be educated on software-intensive system issues and cybersecurity practices sufficient to understand the causes and risks to which these systems

expose the mission or business. For acquisition managers, this education must improve their knowledge of system or product development and assurance practices as well as acquisition practices that ensure the cost-effective delivery of cybersecure systems and products. In government, leaders in the executive and congressional branches must become more aware of the effects of truncated schedules, underfunded budgets, and mismanaged projects on the cybersecurity outcomes of acquired systems.

Using the CISQ Measures for Improving Cybersecurity

The CISQ Measures can be used to control the quality of software-intensive systems. The Security and Reliability metrics should be used in IT management reports and in contracts and service level agreements (SLAs) with vendors and suppliers. The following actions represent best practices for governing and managing cybersecurity risk in software-intensive systems.

- ***Specifying a target risk profile***—management should specify the level of risk that can be tolerated in the operation of a software-intensive system, and any specific source code weaknesses that must be eliminated as part of achieving and sustaining the target risk profile.
- **Describing a system's cybersecurity risk**—the next step in managing cybersecurity risk is identifying exploitable weaknesses in the source code of software-intensive systems and translating the results into a cybersecurity risk statement, such as the probability that a malicious actor could penetrate a system or the extent of the system's attack surface.
- ***Identifying and prioritizing opportunities for improvement within the context of a continuous and repeatable process***—it is critical to supplement traditional testing techniques (functional testing, penetration testing) with system-level static analysis to identify exploitable weaknesses in code and architecture that are difficult to detect through other assurance methods. Executive management must ensure that sufficient effort is dedicated to removing the weaknesses required to achieve the target risk profile for the system. Cybersecurity weaknesses should be prioritized for removal and subsequently removed as part of standard development and maintenance processes.
- ***Assessing progress toward the target risk profile***—the number and type of cybersecurity weaknesses should be analyzed, measured, and regularly tracked as part of accurately assessing progress toward achieving the target risk profile for each system.
- ***Communicating among internal and external stakeholders about risk***— cybersecurity risk and progress toward target risk profiles should be communicated to affected stakeholders through evidence-based measures on the elements creating risk, including valid measures of cybersecurity weaknesses in the source code.

CISQ recommends the following actions be incorporated into development and release practices for critical software-intensive systems.

1) All software security-sensitive, software-intensive systems must be analyzed at each release for critical security or reliability weaknesses contained in the CISQ measures.
2) Identified security or reliability weaknesses in the CISQ measures should be prioritized and incorporated into a backlog of must-fix weaknesses.
3) Measures of the security and reliability of source code should be incorporated as primary contributors to the assessment and measurement of risk in software-intensive systems.
4) Executive management should periodically review the state of their critical infrastructure systems against their risk tolerance, with particular attention to progress on eliminating security and reliability weaknesses.

References

(1) Object Management Group (2015). *Automated Source Code Security Measure*. http://www.omg.org/spec/ASCSM/1.0/PDF

(2) Object Management Group (2015). *Automated Source Code Reliability Measure*. http://www.omg.org/spec/ASCRM/1.0/PDF

Submitters on behalf of CISQ

Dr. Bill Curtis
Executive Director, CISQ
director@it-cisq.org

Tracie Berardi
Program Manager, CISQ
tracie.berardi@it-cisq.org

Joe Jarzombek
CISQ Governing Board Member
Global Manager, Software Supply Chain Solutions, Software Integrity Group, Synopsys
joe.jarzombek@synopsys.com