Towards Formalizing Software Bugs

Irena Bojanova, Yaacov Yesha, Paul Black, Yan Wu

High-confidence systems must not be vulnerable to attacks that would lower the security, reliability, or availability of the system as a whole. That is, they must have no software weaknesses (bugs) that could result in a vulnerability. A vulnerability is "a weakness in system security requirements, design, implementation, or operation that could be accidentally triggered or intentionally exploited and result in a security failure." [1] The Common Weakness Enumeration (CWE) [2] is a collection of known software weakness types described in plain English with demonstrative examples, relations to other CWEs, and some formal definitions. It is a considerable community effort, but many of the descriptions are inaccurate, incomplete, inconsistent, or ambiguous. In this talk, we discuss our vision on precisely defining software bugs. Pure formalization and wholesale restructuring of CWEs is needed to enable automatic generation on software testing components and tools, software analysis tools that always find these weaknesses, formal proofs of systems properties, assuring absence of particular vulnerabilities, and mitigating vulnerabilities by filtering out exploiting attacks. Accurate and precise definitions of software weaknesses are essential for constructing formal proofs of presence or absence of such weaknesses in a program.

We picked three CWEs that we thought would be representative and would help expose the opportunities and challenges of formalization: CWE-307: Improper Restriction of Excessive Authentication Attempts; CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer; CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection'). Why did we choose these CWEs? One justification is their importance. CWE-78 and CWE-307 are among the CWE/SANS Top 25 Most Dangerous Software Errors. CWE-119 is a generalization of CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow'), which is also in the Top 25, as are several other buffer-related CWEs. We also included CWE-119, because we believed it would be relatively easy to formalize. The three selected CWEs are also quite different from each other in aspects from source code faults to ways of exploitation.

In our talk we will present proposed new definitions, contrasting each with the CWE description summary, and then we will explain some of our ideas toward restructuring CWEs and formalization [3]. Our definitions are substantially more precise and accurate than the current CWE descriptions. For instance, the current description summary of CWE-307 includes failure to prevent "multiple failed authentication attempts within a short time frame". However, that description does not provide precise meaning for "multiple" and "short". Our new definition recognizes that CWE-307 actually represents a set of weaknesses, each of which satisfies particular institution specific definitions of "multiple" and "short". Next, the description summary of CWE-119 is "The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside the intended boundary of the buffer". However, "read from or write to a memory location" is not tied to the buffer. Our definition clarifies that access is through the same buffer to which the intended boundary pertains. Our definition also accurately, precisely, and concisely describes violation of memory safety. Finally, the description summary of CWE-78 is based on the software "using externally influenced input", and incorrectly neutralizing "special elements that could modify the intended OS command". "Using input", "intended command", and "correctly neutralizing" are imprecise. We precisely define "using input" and "intended command". We do not include "correctly neutralizing", because it simply means that intended OS command cannot be modified. This work shows why formalization is important and demonstrates that a more rigorous level of definitions is achievable.

[1] Stoneburner, G. et al., "Engineering Principles for Information Systems Security (A Baseline for Achieving Security)", Revision A, NIST Special Publication 800-27 Rev A, June 2004.
[2] The MITRE Corporation, CWE, Common Weakness Enumeration, http://cwe.mitre.org/index.html
[3] Wu, Y., Bojanova, I., Yesha, Y., "Reconstruction of Common Weakness Enumeration", IEEE Software Technology Conference (STC), 2014.