Global Information Assurance Certification Paper

Vulnerability as a Function of Software Quality

Evelyn Labbate

The notion that poor software quality leads to security vulnerabilities in software systems is well known. Call them software flaws, defects or bugs -- it doesn't matter, but the term "defects" will be used herein.

Software defects that cause a program to abort result in denial of service to its end users. Software defects that cause or allow a program to overwrite memory result in buffer overflow conditions, which are commonly known in the security industry to create vulnerabilities in systems. Software defects that allow unauthorized or inadvertent access to system resources or data result in unauthorized disclosure or modification of private information. Therefore, increasing the quality of software will in turn reduce the risk of introducing vulnerabilities into a system.

What exactly is "software quality"? Humphrey states simply that a product that provides the capabilities that are most important to its users is a quality product [1]. The subject of software quality can be taken from many aspects [2], but one can simply hypothesize that a software product should do what it is supposed to do, and NOT do what it is NOT supposed to do. Software is NOT supposed to abort unexpectedly, or inadvertently overwrite memory, or provide a means to allow unauthorized access or exploitation, or output data erroneously, and so on.

Improving software quality addresses the source of many security vulnerabilities, whereas applying procedures and patches to software after exploits are discovered merely addresses the symptoms.

The Road to Software Quality

The requirement for better software quality is easier said than implemented. The business of software engineering and development is a relatively immature industry. Initiatives such as those by the Software Engineering Institute (SEI) strive to promote a sorely needed set of fundamental software engineering standards:

> The SEI mission is to provide leadership in advancing the state of the practice of software engineering to improve the quality of systems that depend on software.
>
> The SEI accomplishes this mission by promoting the evolution of software engineering from an ad hoc, labor-intensive activity to a discipline that is well managed and supported by technology. [3]

For example, the Capability Maturity Models (CMM), developed by the SEI [4], and the ISO 9000 series of standards, developed by the International Organization for Standardization [5], address quality in process and product management.

Today, the information technology (IT) industry is not sufficiently held accountable for the production of defective software products. In the rush to market in a highly competitive environment, the lack of quality controls results in exploitable software and systems. In effect, "we are doing it to ourselves".

Side note: This author has been in the "software development trenches" for years, supporting Department of Defense technology initiatives in a variety of domains. This author also teaches business systems development undergraduate courses (fundamentals of the software development life cycle), and is very familiar with the state of computer science curriculum through academic research. It is this author's professional opinion that the thrust of the American Education System's curriculum in the

information technology areas does not adequately support the tenets of quality software engineering. This aspect is beyond the scope of this paper, however it is a relevant factor in the overall "state of affairs" in the IT industry today.

Following are some suggestions for increasing software quality: from an individual software engineer's point of view, a software team approach, and an organizational perspective. Increased software quality will reduce side effects that contribute to security vulnerabilities that negatively impact the confidentiality, availability and integrity of software and electronic data.

What Individual Software Engineers Can Do to Improve Software Quality

The Personal Software Process (PSP)^{sm}, among other things, induces individual software engineers to monitor and improve their skills at catching potential software defects as early as possible. This book presents a methodology, with worksheets, for using disciplined personal practices to plan and track work in order to increase the quality of software produced. PSP-trained software engineers produce code that, right from the start, has fewer defects [1]. Since effort devoted to defect removal and rework is sharply reduced when defects are detected early, productivity is increased, software quality is increased, and typically, schedules are shortened - faster and cheaper as well as better. In summary, the PSP entails:

- maintaining an engineering notebook for tracking activities and time spent on those activities
- developing product plans, that include:
  - the size and important features of the product to be produced
  - an estimate of the time required to do the work
  - a projection of the schedule
- developing period plans that defines the activities during a certain period
- managing commitments
- managing the individual's schedule in order to meet commitments
- conducting personal code reviews, the principal defect-removal method introduced in the PSP
- gaining an understanding of how defects are injected into the individual's product and process
  - at the design level
  - at the implementation level
- honing skills in software project planning, estimation and measurement.

Humphrey contents that the individual's personal commitment to quality is not a question of whether one can do defect-free work, but whether one cares enough to continue doing it. In his words, "In spite of all the available tools and methods, the most important single factor in program quality is the personal commitment of the software engineer to developing a quality product". [1]

What Software Teams Can Do to Improve Software Quality

Conducting regular code reviews will increase software quality by virtue of the identification of software defects. Code reviews by other software engineering peers, when conducted in a non threatening fashion (i.e. with the understanding the source code is being scrutinized, and not the intelligence or practices of its developer), may uncover incorrect assumptions embedded in the coded logic. Code reviews can also assist in the enforcement of established coding standards (if any!) within the source. Code reviews can also concentrate on specific security-related areas [6] of well-known problems.

One of the tenets of Extreme Programming [7] is the development of code by pairs of programmers working together at a single computer. Pair programming increases software quality because quite simply, "two heads are better than one". While one programmer is typing, the other programmer can catch mistakes or incorrect assumptions. While one programmer is writing the code in syntax, the other can help ensure that the code being written is semantically correct within the context of the logic. From a

security aspect, pair programming instills a "two person integrity" check during source code development. If one of the programmers was intent to introduce potentially malicious code into the source, pair programming would make it that much harder to accomplish.

Establish a configuration management plan and institute configuration management procedures that are enforced by individuals outside of the development team. Otherwise, the fox-watching-the-hen house scenario will inevitably undermine the benefits of a controlled set of baselines.

Utilize an integration and test team with members who possess programming skills, and who are are independent of the software development team. This enables the development of complete and independent test plans and procedures. The test procedures should include a comprehensive set of input variables, those that are expected by the software, as well as extraneous, variable length input strings. Test cases that are constructed by the development team may not thoroughly or accurately test problematic or potentially vulnerable areas of the code.

What Organizations Can Do to Improve Software Quality

"First class people are essential, but they need the support of an orderly process to do first-class work" [8].

An organizational commitment to the resources -- in time, money and people -- necessary for instituting software quality processes is the first step. Senior management buy-in and continued support through the changes needed to achieve, for example, advanced CMM compliance levels and ISO 9000 certification is crucial. Software systems are enabling technologies, and the expenditure of resources into them does not provide organizations with an immediate return on investment. However, the production of higher quality software will, in the long term, reduce the high costs of software maintenance that plague the IT industry. Higher quality software will eliminate future software vulnerabilities.

Conclusion

This paper has established that some security vulnerabilities are a result of software defects. Defects may be injected into software when the environment in which it is developed is not conducive to the construction of high quality software products. A professional commitment, from the software engineer, the development team, and the software organization, is the linchpin in the pursuit of high quality software products.

In summary, the IT industry has provided our society with unprecedented technological innovation. But in doing so, this immature industry has allowed its box of "crown jewels" to become easily penetrateable due to poor software quality. Software organizations must strive to advance their processes and procedures in order to eliminate software defects that lead to software vulnerabilities.

1. Humphrey, Watts S. Introduction to the Personal Software Process[sm]. Reading, MA: Addison-Wesley Publishing, 1997.

2. Quality Attributes. SEI Technical Report. URL:http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.021.html

3. Software Engineering Institute, URL: http://www.sei.cmu.edu/sei-home.html

4. Capability Maturity Models (CMM), URL: http://www.sei.cmu.edu/cmm/cmms/cmms.html

5. International Organization for Standardization, URL: http://www.iso.ch/index.html

6. Security Code Review, URL: http://www.sans.org/infosecFAQ/code/code.htm

7. Extreme Programming, A Gentle Introduction. URL: http://www.extremeprogramming.org/

8. Humphrey, Watts S. Managing the Software Process. Reading, MA: Addison-Wesley Publishing, 1990.

Last Updated: July 20th, 2017

Upcoming Training

SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS