Integrating security into software development

Security needs to form an integral part of the complete systems development life cycle and beyond, from the requirement definition to retirement of application. The objective of secure software development is to design, implement, configure, and sustain software systems with security already built in.

By Anton van Heerden 20 Apr 2012

The security of software is threatened at various points throughout its life cycle, both by intentional choices and actions taken by "insiders" who are producing, deploying, operating, or maintaining the software, and b "outsiders" who have no affiliation with the organisation.

During development, software may be corrupted in ways that will compromise the software's dependability and trustworthiness when it is operational. It is, therefore, crucial to ensure that processes are mature and incorporate security checkpoints throughout the entire system development life cycle. At Altech ISIS we have adopted the Capability Maturity Model Integrator (CMMI) processes and aim always to operate on a level 3 maturity level.

Vulnerabilities may be discovered

Once software has gone operational, vulnerabilities may be discovered and security patches and updates must be applied to incorporate these shortcomings. Any software system that runs on a network-connecte platform has its vulnerabilities exposed during its operation.

The level of exposure will vary depending on whether the network is public or private, Internet-connected o not, and whether the software's environment has been configured to minimise its exposure. The testing cycle should be very rigourous and test cases focussing on security aspects should form part of the overa test pack.

Dependable, trustworthy, and resilient

Software must exhibit three properties to be considered secure: dependable, trustworthy, and resilient. Dependable software executes predictably and operates correctly under all conditions, including when the software comes under attack.

Trustworthy software contains few, if any, vulnerabilities or weaknesses that can be intentionally exploited t subvert or sabotage the software's dependability. In addition, to be considered trustworthy, the software must contain no malicious logic that causes it to misbehave. Survivable or resilient software is software tha is clever enough to protect itself against most known attacks and recover as quickly as possible.

Software is more likely to be secure when security is a key factor in the following aspects of its developme and deployment. The practices used to develop the software and the principles that governed its development are expressly intended to encourage the inclusion of security in every phase of the software' development life cycle.

The programming language, libraries, and development tools used to design and implement the software

are evaluated and selected for their ability to avoid security vulnerabilities and to support secure development practices and principles.

Can the vulnerabilities be remediated?

The software must be expressly tested to verify its security, using tools that assist in such testing. Acquire components must be evaluated to determine whether they contain vulnerabilities and, if so, whether the vulnerabilities can be remediated through integration to minimise the risk that they pose to the software system. The installation configuration of the software minimises the exposure of any residual vulnerabilitie contains.

Finally, the software's analysts, designers, developers, testers, and maintainers must be provided with the necessary information and training to create sufficient security awareness and knowledge to appreciate, a effectively adopt the principles and practices that will enable them to produce secure software.

The key is to understand that software security is not an event, but rather a process that is applicable throughout the software development life cycle and goes on until retirement.

ABOUT THE AUTHOR

Anton van Heerden is the GM of Altech ISIS.