Trustworthy Computing

Vladimir O. Safonov
St. Petersburg State University, Russia

INTRODUCTION

Trustworthy computing is an approach to computer software and hardware development and use aimed to enable security, reliability, privacy, and business integrity of the computing process. The objective of the article is to formulate main concepts of trustworthy computing, to analyze its current status, tools, advantages, issues, and perspectives.

BACKGROUND

Currently everybody's everyday and professional activity is getting more and more dependent on the use of computers and software. So, there is a evidently a growing need in trustworthy computing, since "cyber-attacks" may happen every day, hour, or minute, and can lead to fatal consequences, from a crush of some data or the whole computer system, till a fault of some company or even the death of a live organism somewhat controlled by a computer. The book (Schneider, 1999) considered to be a classical book on trustworthy computing defines this term as follows: "Trustworthiness is assurance that a system deserves to be trusted – that it will perform as expected despite environmental disruptions, human and operator error, hostile attacks, and design and implementation errors. Trustworthy systems reinforce the belief that they will continue to produce expected behavior and will not be susceptible to subversion."

The concept of trustworthy computing is wide and based on a number of scientific, engineering, business, and human factors. This article primarily focuses on trustworthy software development.

TRUSTWORTHY COMPUTING: HISTORY, PILLARS, TOOLS, AND PERSPECTIVES

History

The first computers appeared in the 1940s and their software was inherently non-trustworthy because of their poor user interface, like using punched tape or punched cards to code and input a program written directly in binary machine instructions, and using operator control panel with manual switches and LEDs to input and visualize each bit of memory. However, those computers were "monopolized" by single users, in their turn, for some definite periods of time, so the main danger of user-to-user intervention just could not occur. So a common viewpoint on trustworthy computing is that it originated in the 1960s, with the appearance of multi-user and multi-tasking systems.

One of the first approaches to trustworthy computing was formulated by Allen-Babcock computer company in 1967 (Wikipedia, 2008), in very similar terms as later Microsoft's trustworthy computing approach in 2002: "An ironclad operating system [reliability]; use of trustworthy personnel [business integrity]; effective access control [security]; user requested optional privacy [privacy]."

Later, in the 1960s and the 1970s, computer networking started, which brought many other issues in computing trustworthiness, related to the ability of network attacks, and the corresponding need to mitigate them.

Typical kinds of attacks are: *viruses, Trojan programs, and network worms* – malicious software aimed at making harm to computer systems; *distributed denial of service* – a network attack based on generating lots of

DOI: 10.4018/978-1-4666-5888-2.ch352

requests to server causing its faults; *phishing* – stealing user logins and passwords by frightening and deceiving users with false threats (e.g., threat of deleting their bank accounts) with malicious email messages and Web sites; *pharming* – redirecting users to malicious Web sites to steal their confidential data; *elevation of privilege* – achieving administrator's permissions to modify system files and directories.

Another source of attacks is the use of security vulnerabilities in ordinary software. Classical attack is referred to as *buffer overrun* and uses the following vulnerability. The standard C function of copying strings *strcpy (dest, source)* if often used to copy a null-terminated string *source* to another location *dest*. However, the size of *source* is not explicitly indicated, so it may happen that the result of such copying (either mistakenly or intentionally) will lead to corrupting memory of another process or task. A simple recipe to prevent such attack is the use of a more secure version of the C function – *strncpy (dest, source, length)* where the maximal length of the source string to be copied is indicated explicitly. The book (Howard & LeBlanc, 2002) provides lots of such security recipes.

A historical turning point in trustworthy computing happened in 2002 when Bill Gates (Gates, 2002) sent a historic email to Microsoft employees on the *trustworthy computing initiative*. This event can be regarded as the origin of modern view on trustworthy computing, its principles, discipline, tools, and software development life cycle to support development of trustworthy software products. The immediate reason for that was the need to make an urgent security refurbishment of Windows 2000 after its first official customer shipment, known in Microsoft history as *security push* (Howard & LeBlanc, 2002) - fixing a big number of security vulnerabilities found by Windows 2000 users.

Principles and the Four Pillars of Trustworthy Computing

According to Microsoft's trustworthy computing white paper (Mundie et al., 2002), trustworthy computing should be based on four "pillars": *security* - the ability of software to be resilient to attacks; *reliability* - the ability of software to perform its required functions under certain conditions for a certain time period; *privacy* – preserving personal and company's confidential information; and *business integrity* – on the one hand, prompt way of running software business to enable fixing security, privacy, and reliability bugs, and, on the other hand – correctness of software business. So Microsoft's approach to the problems of trustworthy computing is wide and based not only on purely software development principles but also on business, legal, and human factors.

Security

According to many papers, computer security is a field of computer science concerned with the control of risks related co computer use. In more common sense, security (including computer security) is the ability to be resilient to attacks. In the security paradigm, there are three parties: the *system* to be secured, the *user* willing the security of the system, and the *attackers* trying to break the system. So the main task of the security subsystem in any computer hardware should be to undertake *systematic security measures* to preserve the security of the system, e.g., checking any Web site to be browsed for potential phishing threats; asking the permission of the user to download any kind of document or code (no implicit downloads). Such security actions are likely to make the software running somewhat slower, but this is the necessary security overhead – it is not recommended to switch off any security checks.

In more practical and ubiquitous sense, for *home users*, security means everyday struggle with viruses and worms penetrating into the user's computer from flash memory devices or from the Internet (mostly by email). Their attacks may crush the system or at least are likely to have the user waste a lot of working time for cleaning up the computer from viruses. Another important aspect is the user's understanding how to properly configure security on his/her computer – for example, when using an Internet browser whose new more secure versions may require explicit selection and setting a suitable layer of security.

For *office users,* there are a lot of security aspects to be remembered, including protection of confidential information from malicious actions or from colleagues, securing LANs and WANs, installing and using security patches of the operating systems and software tools of everyday use, etc.

For *software developers*, security should be inherent part of the software process and discipline. A devel-