Trustworthy Computing: A Position Paper

Feiyi Wang (Ph.D.)
MCNC Research & Development Institute
Research Triangle Park, NC, 27709

This position paper addresses two issues: first, what does "trustworthy software" mean to each individual; second, how can a national software agenda be created to ensure that commercial software is of higher integrity? Trust and its associated term of interest, trust worthy software, are indeed very elusive concepts. Trust by definition is "firm reliance on the integrity, ability, character of a person or thing". Trustworthy is something that warrants such reliance, in other words, it has to be earned in someway. Thus, there are two common cases we can label software to be trustworthy: one, I installed a piece of software, it does what it says or as I had expected without a glitch, and I say the software is trustworthy. In another case, I bought *X* software from *Y* company. Company *X* has a reputable name, or referred by another trustee *Z*, therefore, I label software *X* as trustworthy as well.

Note that in both cases, trustworthiness doesn't have much to do with security per se. To take the above argument a bit further for a formal definition, I would like to quote Matt Bishop in his book "Computer Security: Art and Science", which I generally agree: "An entities is *trustworthy* if there is sufficient credible evidence leading one to believe that the system will meet a set of given requirements. *Trust* is a measure of trustworthiness, relying on the evidence provided".

Credible evidence can come in various forms: one can follow a particular development methodology or paradigm (for example, open source development, and many eyeballs' theory), design analysis and formal verification methods, standard practice or the software can be measured against a particular certification process for an imposed completeness and rigor. Ultimately, the sufficiency of such evidence will be judged against the given requirements.

The challenges and difficulties arise when we try to archive trustworthiness in large scale software development, where most of mission-critical applications fall into. I echo the concerns that: each generation of hardware development allows us to create ever larger, more complex systems, but we are also increasingly defeated by that complexity. It couldn't have been said better than an open call from DARAPA, "The raw power that has so seductively invited us to build systems of unprecedented size and complexity has led to systems that are dauntingly difficult to debug and test (thus stretching out delivery times), regularly fail in practices, and are increasingly vulnerable to attack". In short, It is my belief that we are building something we often don't have full grasp, or we think we have but we don't actually, which has eroded every corner of building next generation trustworthy computing software.

With that in mind, there are a lot of things we can do as part of a national research agenda. *First*, more long term research investment. I understand the argument and incentive for sponsoring research of "low hanging fruits" and quick turn around. However, I do believe that there are not silver bullets and shortcuts in addressing such a fundamental challenge. Every step of good, tangible progress requires serious effort and serious time. Often times, those seemingly hard paths one hesitates to take are precisely the right paths to take. *Second*, we need better curriculum in trust computing education. Take computer security course for example, due to its practical importance, IPSEC, WEP etc. are something that one can't miss, while other subjects such as Bell-Lapadula model, noninterference and policy composition, confinement problem, system perspective on information assurance, formal methods for verification etc. become secondary or none at all. Granted, this is a fairly casual observation and there are many good lectures out there, but it is the trend that concerns me. *Third*, we need more awareness and work on cost-benefit analysis, and this is paramount to commercial software development. One has to understand that every security solution, or to the large, trustworthy enhancement effort, come with some forms of costs: performance, usability, financial, or a combination. To justify this cost, there has to be some well-received analysis or even quantifiable results as incentive.

Needless to say, these analysis and observations are quite coarse-grained, but I do hope this can prompt further discussions and be helpful in some way to the workshop.