A short note on secure operating systems, Linux, and the Common Criteria.

©Victor Yodaiken, FSMLabs

The anti-Linux marketing offensive by GreenHills Software and others is based on exactly the sort of murky notions about security that the Common Criteria was intended to discourage. The Common Criteria framework encourages people to define "more secure" against the actual type and role of the software and a clear threat analysis. Clear threat analysis is not served by emotional language, buzzwords, and similar marketing ploys. This note responds to both GreenHills and to a widely cited article that appeared in EETimes based on remarks of Professors Eugene Spafford and Cynthia Irvine [1].

1. Professor Spafford's complaint about the "provenance" of code in Linux's open development model is unfounded. There is no assurance that **any** software development effort is free from people who have bad intent or who just write lousy software. The US government highest security agencies have discovered spies working at the most trusted levels – does anyone realistically expect that software companies will adopt more rigorous screening than the CIA? Fortunately, the Common Criteria suggests solid engineering instead of loyalty oaths and it calls for a rigorous assurance process to track code development and to validate against specifications and threats. "Provenance" is a side issue, one that is easily turned into cheap fear-mongering and xenophobia[2]. For the same reason, I very much dislike the terminology "subversive code" which is an emotionally charged substitute for "malicious code". A security flaw in software developed by the most patriotic and security conscious citizen is no better or worse than a security flaw in software developed by a nefarious enemy from one of those foreign countries that so frighten GreenHills CEO Dan O'Dowd (the idea of a US software company getting nervous about "foreigners" is laughable to anyone who has worked in the industry). The real issues are whether software is designed and tested for security and whether the development process assures certain levels

[1] http://www.eetimes.com/sys/news/showArticle.jhtml?articleID=18901858

[2] http://www.ghs.com/news/20040426_linux.html

of quality and tracks how and when code is put into the system. Knowing whether software is GPL, public domain , or proprietary tells you absolutely nothing about how well the program has been protected from malicious code. In fact, because Linux code is developed on an open model and is tracked by a comprehensive source control system (see www.bkbits.com) it may be relatively harder to smuggle malicious code into Linux than into some commercial operating systems.

2. The state of the art should not be over-estimated. When, according to EE-Times Professor Irvine contrasts Linux with “ "high-assurance" operating systems with the smarts to prove that subverting code doesn't exist” we should understand that there are zero existing operating systems that can prove that they don't contain malicious code or other security flaws. In fact, there is considerable disagreement among researchers over best methods, a shortage of empirical data and limits to what can be verified at the highest level. Here's what the NIAP says about EAL7, the highest level of security assurance in the Common Criteria:

   > At the top, EAL7, level there are significant limitations on the practicability of meeting the requirements, partly due to substantial cost impact on the developer and evaluator activities, and also because anything other than the simplest of products is likely to be too complex to submit to current state-of-the-art techniques for formal analysis.
   >
   > (http://niap.nist.gov/cc-scheme/cc_docs/cc_introduction.pdf)

   Vendors that claim their operating systems are 100% validated against credible threats are either lying or announcing a major breakthrough. Ask for a written warranty if you are not sure.

3. The Common Criteria standard defines “evaluation assurance levels” (EAL) from 1 to 7, with 7 being the highest. These levels are being used in a grossly misleading manner. The EAL is just a measure of the level of effort and rigor put into proving that a software program satisfies a security specification. That is, the EAL tells us something about how a program P was validated against a specification S. If specification S does not identify the actual threats to the system, the software is not secure, no matter what the level of evaluation assurance. A rigorous proof that the Titanic is unsinkable in the Caribbean Sea is no comfort on a voyage through the North Atlantic iceberg belt. A program that is EAL 7 validated against a specification that says nothing about real-time is no

more secure against threats to real-time behavior than a program that has not been tested at all. GreenHills is currently embarked on a EAL6 (not 7) effort against a specification that does not have a single real-time requirement. If and when the validation effort succeeds it will say nothing about whether the software, for example, can maintain real-time schedules in the face of a flooding denial-of-service network attack.

4. The higher EALs are not ironclad, but they are very costly and possibly impractical. Even EAL7 only requires a "semi-formal" specification of the low level implementation so there is no mathematical proof that the low level specification matches the implementation. In any case, the differences between "formal", "semi-formal", and "informal" are not defined in any rigorous manner and for good reason. Formal specification of complex software is a long way from being a solved problem and it is not even clear that the processes called for in the Common Criteria work. There are no existing EAL7 certified operating systems – not a single one (except maybe hidden in some NSA lab) so it is not known whether, in practice, an EAL7 certified OS is actually more or less secure than other software.
5. Common Criteria is a "best practices" document that is dominated by concerns of governmental security agencies. The standard is relatively new and untested. It seems as if these practices should produce more secure software, but we don't know for sure. Experts like Bruce Schneir argue that standards themselves are of limited use. (http://www.schneier.com/crypto-gram-0105.html). At FSMLabs we make a serious engineering effort to protect our real-time kernels against a variety of attacks that seem critical to control systems. We use many of the practices recommended in Common Criteria, but we have not yet seen the utility in going through the standards process – particularly since there is no reasonable Common Criteria security specification for real-time operating systems.
6. Despite Professor Spafford's complaints about the intrusion of mere cost considerations into software purchase decisions, in the real-world resources are limited and trade-offs are inescapable. Developers will limit functionality in an effort to limit the costs of certification or just to make certification practical. If a limited certified operating system causes the complexity of applications to increase and the reliability of those applications to decrease, use of that software may have a negative effect on the security of the whole system. Is it more or less dangerous

to use Linux to control a power plant than it is to use an EAL7 (say) OS? Suppose the EAL7 OS comes with no device drivers, costs enough to reduce the amount of test time that can be used in development or the number of trained operators used to monitor the plant, and requires application developers to produce their own math library. Suppose the Linux system is not connected to the network. Suppose the EAL7 evaluation is against a specification that does not cover the most likely threats. The answer is: you better do a real whole systems security analysis instead of relying on buzzwords.

7. Finally, I want to point out that the mere existence of a "certified" version of some company's operating system does not mean anything about the other software produced by that company. For both Common Criteria and the FAA's DO-178 reliability certification, the general practice is to set up a separate development team and a separate, limited, product line. The Common Criteria documents even identify EAL4 as the highest level of assurance possible on a "retrofit" to an existing system - making it very unlikely that a EAL7 product is in the main product line. Even the military projects that GreenHills CEO Dan O'Dowd cites, often are not able to bear the costs and/or the limitations of the certified product lines and so purchase the standard commercial versions which receive PR benefits, but not necessarily any reliability or security benefits, from the certified product lines. The interesting comparison between Linux or some other solution is to the actual products being used, not the highest assurance component sold by the vendor.

Software security requires strong engineering and solid cost/benefit analysis, even though that is probably not the best marketing tactic and it means we have to admit that there are no magic bullets to make the problem go away.