Testimony before the House Government Reform Committee Subcommittee on Technology, Information Policy, Intergovernmental Relations and the Census

Exploring Common Criteria: Can it Ensure that the Federal Government Gets Needed Security in Software?

17 September 2003

Statement of
Eugene H. Spafford
Professor and Director
Purdue University Center For Education and Research in Information Assurance and Security (CERIAS)

Co-Chair of The U.S. Public Policy Committee
of The Association For Computing Machinery (USACM)

Member of the Board of Directors
of the Computing Research Association (CRA)

Table of Contents

Introduction

Thank you Chairman Putnam and Ranking Member Clay for the opportunity to testify at this hearing. It is clear that there is a large and growing problem with the security of our cyberinfrastructure. Nowhere is this more apparent than in the computing systems used within the Federal government. Many of us working in information security have been alarmed for years by unfortunate trends in the way software is produced, acquired, deployed, and then used. High on the list of concerns has been the continuing poor quality of software, and in particular COTS (Commercial Off-the Shelf) software. This committee is to be commended for the series of hearings that it is holding on these issues.

This particular hearing poses the question "Exploring Common Criteria: Can it Ensure that the Federal Government Gets Needed Security in Software?" I will explain that the answer to that question is "no."

By way of introduction, I am a professor of Computer Sciences at Purdue University, a professor of Philosophy (courtesy appointment), a professor of Communication (courtesy appointment) and the Director of the Center for Education and Research in Information Assurance and Security. CERIAS is a campus-wide multidisciplinary Center, with a mission to explore important issues related to protecting computing and information resources. We conduct advanced research in several major areas, we educate students at every level, and we have an active community outreach program. CERIAS is the largest such center in the United States, and we have a series of affiliate university programs working with us in Illinois, Iowa, North Carolina, the District of Columbia, Ohio, Virginia, and New York State. CERIAS also has a close working relationship with over a dozen major commercial firms and government laboratories.

In addition to my role as an academic faculty member, I also serve on several boards of technical advisors, including those of Tripwire, Arxan, Microsoft, DigitalDoors, Unisys, and Open Channel Software; and I have served as an advisor to Federal law enforcement and defense agencies, including the FBI, the Air Force and the NSA. I am currently a member of the Air Force Scientific Advisory Board, and I have been nominated for membership on the President's Information Technology Advisory Committee. I have been working in information security issues for 25 years.

I began this document by listing my affiliations with ACM and CRA. This testimony is not an official statement by either organization, but is consistent with their overall goals and aims. ACM is a nonprofit educational and scientific computing society of about 75,000 computer scientists, educators, and other computer professionals committed to the open interchange of information concerning computing and related disciplines. USACM, of which I serve as the co-chair, acts as the focal point for ACM's interaction with the U.S. Congress and government organizations. USACM seeks to educate and assist policy-makers on legislative and regulatory matters of concern to the computing community. The Computing Research Association is an association of more than 180 North American academic departments of computer science and

computer engineering, industry and academic laboratories, and affiliated professional societies. The CRA is particularly interested in issues that affect the conduct of computing research in the USA. Both organizations stand ready to provide expertise and advice upon request.

Recent events, including the so-called Sapphire, Blaster, and SoBig worms have only served to underscore the vulnerability of our computing systems. Defacement of WWW pages and the spread of annoyance viruses pales in comparison to the potential for damage suggested by news that recent incidents may have affected some of our power systems during the August 15th blackout, that banking networks were unavailable because of contamination, and that sensitive military and law enforcement computers were also affected. As we increase our reliance on computing and networks, our potential vulnerability also increases. Deployment of new technologies, such as replacing some of our telephony with voice-over-IP (VoIP) suggests new reasons to be concerned about computer and network vulnerabilities.

Examination of many of the attacks, tools, and incidents that have come to light over the last few years indicate that far too many of the incidents have been the result of flaws in deployed software. What is even more discouraging is that so many of these flaws are likely the result of carelessness.

My research center (CERIAS) maintains a database of vulnerability reports and patches: The Cassandra Service.[1] On September 14, I performed a search of Cassandra. I selected the 100 software products with the largest number of reported vulnerabilities between January 1, 2000 and September 14. There were 2255 vulnerabilities reported for those 100 products in that time period — an average of over 11 per week. Overall, Cassandra has entries for over 5000 total vulnerability reports. Some of the programs had over 100 flaws reported in this period — an average of one every two weeks. Based on prior experience, there are undoubtedly many more flaws that have yet to be found, or that have been found and not yet reported to authorities

Analysis by my research staff earlier in the year revealed that over 20% of the reported flaws archived in Cassandra were caused by the failure to properly check for bounds on buffers, resulting in the possibility of attack via a "buffer overflow." This is when an attacker provides more input than the program was coded to accept, and the programmer failed to properly account for the extra input. The result is that arbitrary data can be sent to overwrite program code or control information, leading to exploitation of the system. Analysis of data in NIST's ICAT database[2] provides a similar figure, with 22% of the reported flaws being buffer overflows.

This is discouraging as buffer overflow is one of the first things we teach our students to avoid when they take their beginning programming classes. It is simple to avoid. It is a flaw that we have known about as a threat to both reliability and security for over 30 years. The infamous Morris Internet Worm of 1988 exploited a widely publicized buffer overflow, and, sadly, 15 years later we are still seeing commercial software written with buffer overflows. Vendors are

[1] This is named after the Trojan woman of legend who was cursed by the gods to have the power of prophecy but to never be believed. She warned the leaders of Troy not to take the wooden Trojan Horse inside the walls, but they did not listen to her. Troy was destroyed and Cassandra killed as a result. Cassandra access is free, and is available at <https://cassandra.cerias.purdue.edu>.

[2] See <http://icat.nist.gov/icat.cfm?function=statistics>

failing to learn from the past.

Further analysis on Cassandra data revealed that another 27% of the reported flaws were from other forms of failure to validate input (e.g., check bounds or correctness of values), and that other forms of simple design error accounted for an additional 26% of the flaws. Thus, nearly 3 out of every 4 reported flaws was the result of programmers making simple mistakes that have well-known causes and have been known and taught about for several decades.

It is clear that much of the software being used today in government and industry is severely flawed. What is more, the pressure of the marketplace has done little to eliminate these flaws. In some cases, it might be conjectured that the pressure of the marketplace has actually exacerbated the problem:

- Pressure by consumers for new features has led vendors to increase the complexity of software beyond the point where it is understood or completely testable. Companies with products having fewer features and reduced complexity are penalized as the population has sought out products with additional features even if they are not needed.
- Pressure for a reduced time-to-market has led to products being rushed in design and production, and then shipped before adequate testing has been performed. Companies with slower release cycles are viewed as "not innovative enough" and penalized in the marketplace.
- Efficiency of scale has resulted in vendors producing monolithic offerings that contain all features and options rather than specialized versions tailored for individual markets. It is not at all clear that the same software should be deployed in both environments. Consider that the same PC operating system, text processing system, spreadsheet, WWW browser and database system is likely to be found running on systems that contain and protect our nuclear secrets and intelligence information, and also on a home PC with someone's recipe book and baby pictures.
    - ◇ This results in reduced cost for the vendor, requiring only to produce, package, and document a smaller number of systems.
    - ◇ This results in reduced purchase cost (but not necessarily reduced operational cost[3]) for large customers as they can use the same hardware platform, add-on products and user training throughout the enterprise, despite wide variation in use of the product.
    - ◇ This increases the opportunities for an attacker because he can obtain and test attacks against the same software that is running in sensitive applications.
- Increased testing and better software engineering methods require hiring better-trained personnel, employing more expensive tools, and spending more to assure quality. End users have not shown a willingness to pay extra in support of this quality, thus putting companies at a disadvantage if they increase the care with which they develop code.

[3] Many firms and government organizations separate operational costs from acquisition costs. Thus, a system that is $100 cheaper per seat, but requires $1000 more per year in after-market patching, anti-virus software and helpdesk support is often purchased as a means of "saving money."

The lack of any meaningful penalty for producing flawed software (e.g., liability torts) has meant that there has been little in the way of normal market pressures to counteract the above.

The same factors may be indirectly affected academia's ability to train better software engineers. If a college or university were to commit to teaching its students more stringent software engineering practices and the use of more reliable programming languages, those students would not be as attractive in the marketplace: they use methods that are viewed as more expensive and less portable than current practice. Before long, the college would find it would no longer be attracting enough students to continue its program. Furthermore, corporate donations might well be reduced, increasing the cost of the program to maintain.

Similar pressures would — and do — impact research. Government and industry funding seems to be heavily oriented towards new and more efficient patching and protecting existing systems. This is not unexpected because so much has already been invested in those systems. However, that limits our ability as researchers to investigate new and perhaps more robust architectures and approaches, and thus serves to perpetuate some of the same flaws that are haunting us today.

The Common Criteria: Pluses and Minuses

Over the course of the last 30+ years there have been a series of standards and certifications for software quality assurance. DOD-STD 5200, the TCSEC, ITSEC, Federal Criteria, FIPS-140 and others have each attempted to address the need for strong protective measures in IT for government use. Each has had successes and flaws. The Common Criteria (CC) effort is the latest in this line of standards, and draws from the experience gained with those other efforts. It contains a number of much-needed enhancements over previous efforts, and has achieved widespread acceptance around the world.

For purposes of this testimony, I will assume that the Committee has had some form of tutorial on the Common Criteria and is familiar with its basic structure and terminology. As such, I will only address some major points about it here.

The Common Criteria is intended to provide a formal way to describe and evaluate the security properties of a software artifact. This has value when trying to understand the particular strengths of a product. However, the Common Criteria has a number of drawbacks that have been noted by those who have studied it. If we were to consider the impact of requiring Common Criteria evaluation for all software sold to the government, many of the disadvantages come to the fore.

- Malicious code hidden in software, such as might be introduced in offshore coding houses or by domestic criminals, is difficult for even trained auditors to find. Certification at any level below EAL-6 would probably not find such additions, and a clever programmer might still be able to hide some inside code that would evaluate at the highest level: EAL-7.
- Certification is oriented towards evaluation of products, not working systems. Thus, combining products together as occurs in normal operation may result in a system that is not certified, and may actually result in a system with weak overall security.

- Certification under CC is heavily weighted towards examination of documentation and vendor-produced artifacts. The lack of detailed, 3rd-party testing can be viewed as a significant weakness in the evaluation process.
- There are too few well -articulated protection profiles (PP) defined against which security targets (ST) can be written and products evaluated. To be useful, PPs should be written for specific environments and take into account a full risk assessment, and this is difficult to do. For this reason, PPs are difficult to develop and certify. PPs that are too general are not likely to be helpful. PPs describing unrealistic environments are similarly unhelpful.
- Certification for systems in high-risk environments, which are many of the government systems of most concern, should have certification at level EAL-5 and above. No certifications have been done at these levels, nor are the standards yet defined to support performance of such an evaluation.
- Although some systems might require evaluation at EAL-6, other systems would not require certification at such a high level because of their usage or risk level. Defining the appropriate level and protection profile needed for each application would be very time and labor intensive.
- Certification is currently expensive and time-consuming. Requiring CC certification of products might well keep them out of government use for years while the commercial sector is using the more recent versions.
- The government market is small enough in some sectors that vendors may choose to forego selling to it rather than undergo the time and expense of certification. This could prevent government agencies from taking advantage of new, more capable software and from interoperating with widely-used commercial software. This was often the case in the 1970s and 1980s under previous certification regimes.
- Software could be required to be certified at a higher level than necessary. For the vendors to comply, the cost of development and testing would need to be borne by the customers and end-users of that software, thus resulting in higher prices.
- There are not enough CC certification facilities to handle the load of products to be tested. Specialized training is required to produce qualified personnel to staff such facilities so there would likely be a severe backlog of products awaiting certification, exacerbating several of the other problems described in this list.
- The CC fails to consider standard product lifecycles. As such, products are often out-of-date by the time they are certified, and may not be the most mature, tested version of the software. The higher the evaluation level, the larger this lag becomes.
- Although there are mechanisms for certifying code that undergoes patching and after-market customization, it is not obvious if software with extensive patching needs and customization will be adequately covered under this mechanism. Current software does undergo frequent patching and upgrades.
- New, experimental and boutique products currently gain market traction by showing their

worth in controlled trials. If CC certification is required, the time and expense may keep these products from fair trial, and thus prevent them from succeeding in the market. This also includes university prototypes and experiments that may be deployed in new and novel manners. Many of the most important security tools in use in the market today would not be considered because they had their beginnings as small, leading-edge products.

- There is no obvious provision for open source software to be certified using the Common Criteria. The cost of certification and the required effort to produce an STare both prohibitive. Althought open source is not better, per se, for security, there are some open source products that may be more appropriate for use in some circumstances. The current approach to CC certification would almost certainly exclude those products.
- Even certified code can support and spread viruses and other malware.
- Even certified code can contain flaws that can be exploited by attackers.

To the positive, however, the discipline brought to development to support eventual Common Criteria evaluation can be valuable. If more vendors actually did target risk assessments, thought about attacks and defenses, developed formal requirements documents, ensured that documentation was correct and up-to-date, used good production methods, etc, then our software infrastructure would be greatly improved. In fact, it is the *process* of software development rather than the *product* that may be most important in quality assurance. A quality process is more likely to result in a quality product. This fact is well-known in software engineering, and in engineering in general, but seems to be given little attention in many commercial software settings.

As an illustration of some of the issues, consider Microsoft's Windows 2000 product[4]. It was certified in October of 2002 at EAL-4+, which is currently the most rigorous certification available. However, this same system was the target of the Blaster worms, has been victimized by dozens of viruses, and has been the subject of several dozen patches for serious vulnerabilities discovered since the time of its certification. Although it is undoubtedly a safer product than it would have been without the effort to have it certified, it is certainly not what most people would consider a "secure" system. The value of the certification is therefore unclear.

Another aspect of this certification that should be noted is the protection profile (PP) against which Windows was tested. The PP defines the usage and threat model, and the choice of PP significantly impacts the interpretation of any certification results. Windows 2000 was certified against the CAPP protection profile. Dr. Jonathon Shapiro has posted an essay on his WWW site (<http://eros.cs.jhu.edu/~shap/NT-EAL4.html>) that addresses the certification of Windows 2000 at the CAPP/EAL-4 level. The following is a quote from his essay:

The Controlled Access Protection Profile (CAPP) standard document can be found at the Common Criteria website. Here is a description of the CAPP requirements taken from the document itself (from page 9):

> The CAPP provides for a level of protection which is appropriate for an assumed non-hostile and well-managed user community requiring protection against threats of inadvertent or casual attempts to breach the

[4] This is not meant to suggest that Microsoft's products are any better or worse than that of other vendors. This example was chosen to represent a product widely used within government and industry.

> system security. The profile is not intended to be applicable to circumstances in which protection is required against determined attempts by hostile and well funded attackers to breach system security. The CAPP does not fully address the threats posed by malicious system development or administrative personnel.

Translating that into colloquial English:

> Don't hook this to the internet, don't run email, don't install software unless you can 100% trust the developer, and if anybody who works for you turns out to be out to get you you are toast.

In fairness to Microsoft, CAPP is the most complete operating system protection profile that is presently standardized. This may be the best that Microsoft can do, but it is very important for you as a user to understand that these requirements are not good enough to make the system secure . It also needs to be acknowledged that commercial UNIX-based systems like Linux aren't any better (though they are more resistant to penetration).

Dr. Shapiro also notes that if he were to write a program that did nothing but paint the screen black (or blue), then it would be possible to have it certified at an EAL-4 or even EAL-7 level. All that the certification would mean is that the program reliably paints the screen, as designed and documented. There is really no conclusion that can be reached about the underlying fitness for purpose or utility of the software.

The Environment

The environment in which we currently deploy computing adds to many of our problems. We have a large base of legacy software and hardware that cannot be replaced quickly. In some cases, we would face incredible difficulty in replacing key systems, as we discovered during the search for Y2K problems. This means that any solution that applies to future systems yet to be produced will only gradually have an impact on the overall problem.

It is the variety of systems and operational environments that adds yet another layer of concern to security. We have such complex interactions and interconnections we cannot begin to understand all of their nuances. Years ago, my students and I described a series of security flaws that we labeled as “emergent faults” in our research. These come about when multiple systems, each operating correctly and securely according to their design, interact with each other and with the environment in unexpected ways to produce a failure. No single thing has gone wrong, but the combination has resulted in catastrophic failure. Even if several software artifacts are certified as Common Criteria compliant, once they are put together on a system or network it may be possible for them to interact in unexpected manners, leading to cascading problems.

We must also consider the effect of accidental physical failures. If there is a power failure in a government building, is it possible that a security monitor or firewall shuts down first in such a way as to leave other machines unprotected? Is it possible that a disk failure can result in an audit trail being lost? What of a lightning strike inducing current in a network line, a broken water main spilling water into a large data storage server, a failure of air conditioning resulting in overheating and failure of the main VoIP switch? In each case, the failure of the system is one that can lead to loss of data or processing and the presence of Common Criteria certified software does not prevent the damage.

There is also the case of physical failures from malicious actions. Arson and theft are two

malicious acts that can have serious impacts on sensitive computer systems. The theft of a diskdrive with personnel information, law enforcement information or building plans could all be catastrophic, but certification of software with Common Criteria would not necessarily mitigate the problem; if the underlying protection profile did not specify strong encryption for the stored data, there might well be little in the way of safeguards on the stolen data. An attacker could also cause the loss of power or water line break described above, either as a destructive act, or to disable a particular set of mechanisms to enable some other activity. The CC will not provide protection against such acts.

The Human Factor

It is a basic fact that we would not have computer security problems if there were no people. Although that may sound flippant, the point is that human beings are the ones who use our IT systems, and humans are the ones who abuse them. We need to pursue approaches that reduce the risk from both approaches.

Humans abuse computer systems by breaking into them, committing fraud, causing denial of service, writing viruses and other malware, committing identity theft, and committing a long list of other criminal activities. No matter how good the technology may be, there will likely always be ways to abuse it. If nothing else, "insiders" who have access to the computer systems are in a position to misuse their authority to access or change sensitive data. We have many historical accounts of traitors accessing national defense information for foreign powers, of law enforcement agents accessing details of confidential informants and undercover operations in return for bribes, and of government officials accessing personal data for inappropriate uses.

Technology can only provide protection for information up to a point. Once the information is accessed by people and other applications with appropriate authorization, it is necessary for non-software methods of protection to be applied. Thus, personnel security mechanisms must be in place to appropriately screen individuals before they are placed in positions of trust, and to periodically reevaluate them. Operational security (OPSEC) methodologies should be applied in the maintenance and operation of the systems to audit usage, detect questionable behavior, and respond appropriately.

One of the most important aspects of information protection is law enforcement. There needs to be a credible threat of discovery and prosecution in place to deter individuals who are considering misuse of IT systems, and to appropriately punish those individuals who do commit transgressions. This requires trained investigators, adequately-equipped laboratories, and the necessary personnel and financial resources to pursue investigation and prosecution.

The current state of law enforcement for cybercrime is far below the level needed to provide an adequate deterrent to criminal behavior. Although there are a number of well-trained investigators at the Federal level, their numbers are far too few to deal with the many cases brought to their attention…or the many more that would be brought to their attention if the victims had some hope of the cases being pursued. There are also a few well-equipped forensic laboratories in the

US, but they are few in number and heavily loaded with casework. There are no nationally accepted standards of training for those examiners, there are a limited number of software tools to use in investigation, and there are few resources being expended to advanced research and training in the area. The cost and complexity of investigation and prosecution is such that officials are often reluctant to pursue cases without proof of a large, obvious loss.

The situation at the state level is even more dismal, with only a few states possessing some advanced resources that can be applied to computer crime investigation. When one considers that many Federal crimes are first discovered as crimes against local entities, and that many offenders are juveniles who are unlikely to be prosecuted at the Federal level, the need for adequate state resources becomes more acute.

The other aspect of human behavior that leads to IT system compromise is that of non-malicious activity brought about by stress, carelessness and/or ignorance. Humans who operate the IT systems may not have sufficient training or resources (particularly, time) to appropriately operate the systems in a safe manner. This becomes a major issue when every physical desktop holds a high-end, networked computer running a complex and (probably) flawed operating system. Small errors in configuration of a single networked system can result in catastrophic, cascading failures of other systems in the enterprise. The ease with which new software can be downloaded and executed (either intentionally, as a browser plug-in, for example) or unintentionally (as occurs with many email viruses) is one source of many security problems. A system certified to a high level in the Common Criteria can still fall prey to these problems unless they are all anticipated and addressed in the design (and in the security target, ST).

Currently, secretaries and low-level administrative personnel are typically equipped with high-end computers running full operating systems capable of spreading viruses and denial of service attacks, when all they may really need is access to a mail program, address book, and WWW browser. The mix of extra functionality combined with the lack of training provides a potent — and dangerous — combination. A more accessible solution than trying to ensure the quality and safety of all the software running on every platform is to explore systems with more limited functionality, including use of "thin-client" systems that remove the main computing platform and unnecessary utilities from the reach of the inexperienced user.[5]

It is also the case that in many environments the administrators of systems are charged with the security of the systems they administer, but they are not given training or support for those tasks. The result is that they are under time pressure to install fixes, configure security tools, follow up on incidents, and respond to user concerns in addition to their regular duties. Without adequate training and tools, they cannot respond effectively nor in a timely manner. Although well-intentioned, they end up letting some critical tasks (such as patch application) slip because of a lack of time.

Coupled with all of this is the overall problem of user interface. Too many products are

[5] A thin-client is basically a terminal or limited workstation connected to a larger computer system. This architecture also brings other benefits, including great ease of administration, more complete archiving of software, easier patching, better control over software licensing, and greater resistance to viruses.

provided without easy-to-understand (and easy-to-operate) controls for the protection mechanisms that are in place. To enable a built-in firewall, for instance, might require as many as a dozen operations to set various options and variables. Not only must those settings be found, but they must be understood — what is the indirect effect of each setting, and what else needs to be configured? Too often the documentation is unhelpful, incomplete, or incorrect. Only some of this is addressed by the Common Criteria process, and even then there is no formal process of determining a match of the user's abilities with the user interface of the security mechanisms. It is little wonder that so many computing systems are configured incorrectly and dangerously.

Some Recommendations

There are several actions that can be taken to reduce the threat of abuse of government computers. All of these can be derived by examining the problems that confront us. These are independent of the Common Criteria. Among those that have the highest likelihood of making a difference, I would include:[6]

1. Emphasize the need for a systems-level view of information security. Assuring individual components does little to assure overall implementation and use. This requires trained personnel with an understanding of the "big picture" of IT security. Too often those who design and specify the systems do not understand how they are actually used....or misused.
2. Establish research into methods of better, more affordable software engineering, and how to build reliable systems from untrusted components. 15-20 years ago the decision was made to cede research in this arena to the commercial sector, believing the market would drive innovation. That has not happened.
3. Increase the priority and funding for basic scientific research into issues of security and protection of software. Too much money is being spent on applying patches to intrinsically unsound systems and not enough is being spent on fundamental research by qualified personnel. There are too few researchers in the country who understand the issues of information security, and too many of them are unable to find funding to support fundamental research. This is the case at our military research labs, commercial labs, and at our university research centers.
4. Explicitly seek to deploy heterogeneous environments so that common avenues of attack are not present. This *may* require some extra expense *at first,* but eventually it may lead to increased compliance with standards, increased innovation, and increased choice in the marketplace, thus lowering costs while increasing security. If real standards (rather than de facto standards) are developed and followed, interoperability should not be a concern.
5. Complementary to the previous recommendation is giving thought to different architectures. Rather than a computer on each desktop, thin-client technologies based on a mid-size computer in a centralized location can provide all the same mission-critical

[6] I provided a similar list to the House Armed Services Committee Subcommittee on Terrorism, Unconventional Threats and Capabilities in my written testimony of 24 July 2003. Although this document does not address all of these points, I believe they are still worth considering.

services, but remove many of the dangerous aspects of distributed PCs. For instance, patches need only be applied in one location, and there is a greatly reduced possibility of untrained users loading untested media or software.

6. Rethink the need to have all systems connected to the network. Standalone systems may not receive all of the latest patches as soon as they come out. However, that alacrity may not be needed as those systems can no longer be attacked over the network.
7. Require greater efforts to educate personnel on the dangers of using unauthorized code, or of changing the settings on the computers they use. It is still often the case that personnel will turn off security features because they feel it slows them down or gets in their way. Unfortunately, this can lead to significant vulnerabilities.
8. Revisit laws, such as the DMCA, that criminalize technology instead of behavior. It is extremely counterproductive in the long run to prohibit the technologists and educators from building tools and studying threats when the "bad guys" will not feel compelled to respect such prohibitions.
9. Provide increased support to law enforcement for tools to track malware, and to support the investigation and prosecution of those who write malicious software and attack systems. This includes support for additional R&D for forensic tools and technologies.
10. Do not be fooled by the "open source is more secure" advocates. Whether source is open or proprietary is not what makes software reliable. Rather, it is the care used to design and build it, the tools used to construct and test it, and the education of the people deploying it. In fact, some Linux distributions have had more security flaws announced for them in the last 18 months than several proprietary systems. However, some open source software, such as OpenBSD and Apache, appear to be far more reliable than most proprietary counterparts. There is no silver bullet for problems of quality and security, and that includes the Common Criteria.
11. Initiate research into the development of metrics for security and risk. Acquiring systems based on cost as the primary criterion is not reasonable for mission-critical applications. We need to be able to differentiate among different vendor solutions, and set standards of performance. Common Criteria evaluation is not sufficient for this purpose, especially when systems are evaluated against very different protection profiles.
12. Establish better incentives for security. The current climate in many government agencies is to penalize operators for flaws, thus leading many of them to dread enhancement and exploration of better security.

Conclusion

. It is clear that we have deficiencies in our cyber defenses. Poorly designed and incorrect software poses a particular threat because it can be so widely deployed in government and the civilian sector. We need to find better ways of increasing the quality of the systems we purchase and deploy. For the reasons given in this testimony, application of the Common Criteria cannot ensure that software used by the Federal government will provide a sufficiently secure

environment. The following quote[7] captures a basic truth about security and certification:

> Many techniques exist for obtaining assurance. Whatever technique(s) is used, a fundamental point is that quality comes from the producer of the system, not from subsequent testing and evaluation. Testing and evaluation provide insight about the level of quality that exists, rather than defining a level of quality; i.e., reason for confidence that the information security will work correctly and effectively. If the system has the necessary quality, then testing and evaluation can provide information about the quality. However, if the system lacks necessary quality, then testing and evaluation cannot make up deficiencies in quality in a complex system.

Good security is not something that is an add-on or after-the-fact concern. Methods other than the Common Criteria will need to be employed to address critical security needs, and those methods will require a sustained investment of resources and research to have maximum effect.

I will be happy to expand on any of these points, now or in the future.

Thank you again for the opportunity to provide this testimony.

Acknowledgments

I received many suggestions from colleagues when composing this testimony. I wish to acknowledge the people listed for their assistance. However, the content and opinions expressed are my own, and the presence of these names should not be construed as endorsement of any of the statements herein.

James Foley, Simson Garfinkel, Peter Harsha, Ben Kuperman, William Malik, Robin Roberts, Marv Schaefer and Paul Williams.

[7] From *Security Considerations in the Information Systems Development Life Cycle* by Timothy Grance and Marc Stevens. To appear as NIST publication 800-64 later this year.