The Evolved Threat Paradigm: Look Who's Wearing the Black Hats!

Dixie B. Baker
The Aerospace Corporation
El Segundo CA 90009

Abstract

The "average user" against which most of the classes of the Department of Defense Trusted Computer System Evaluation Criteria [1] (TCSEC) were developed is much more computer literate, and perhaps more "sophisticated," than was envisioned. The threat against which the higher-assurance classes (B2-A1) of the TCSEC were developed – the "malicious user" – has emerged as far less nefarious than was assumed. Computer technology is far more powerful, distributed, and mass-marketable than could ever have been predicted. Yet the standard developed ten years ago for building trustworthy systems remains with us, and few systems are being designed to meet its most stringent assurance requirements. This paper challenges the appropriateness of this legacy in our current, rapidly evolving threat environment, contends that the TCSEC classes for which products are being built provide user services but little assurance, and suggests that a "security infrastructure" may be needed to accommodate the new "bad good guy" paradigm.

1 Background

When the TCSEC was first published as CSC-STD-001-83 [2], on August 15, 1983, few people questioned who the "bad guys" were. For those with doubts, President Reagan could very quickly provide the needed clarification. The people knew that the "bad guys" were those foreign superpowers who threatened Democracy and human rights; those who were building massive arsenals of nuclear weapons capable of global destruction; those who overtly declared their intent to overpower these United States of America.

On the day CSC-STD-001-83 was published, the *New York Times* contained an article entitled "Computers Draw 1,100 To Classes." This article described an innovative summer program ("computer literacy camps") in Queens, the Bronx, and several other city high schools where "children are learning about microcomputers, data processing, computer programs, and Cobol, the language of computers." One 14-year-old student, sitting in front of her TRS-80 Model 4, stated that her motivation for participating in this voluntary program was: "I want to get a good job and stuff, and computers are really in." And a 13-year-old made the astute observation that "Computers are really coming into the world." [4]

Not everyone agreed with this 13-year-old, however. The *Wall Street Journal* on that day contained a "Manager's Journal" editorial by Jack Falvey entitled "Don't Count Too Heavily on That Personal Computer" [5]. Falvey noted that "The personal computer has now become the status symbol of our times," comparing it to the earlier hoola-hoop craze. His view was that the personal computer "lack(ed) a clear-cut purpose." Personal computers to him were "a great answer in themselves looking for a great question." He noted that business problems were not "repetitive," which was the only thing personal computers were good at, and that bookkeeping was best done by a bookkeeping company. He warned that computers "don't solve business problems," and "as a matter of fact, they create a whole new set of problems–not the least of which is learning to trust a segment of your business to a computer and its support system over which you may have only very limited control at best." But his real *coup de grace* was:

> All of these issues pale in the light of one major fact. Up until a few months ago, most small computers either were toys used mostly for games, or they were industrial products marketed in bulk or as systems. The computer isn't literate yet, and it doesn't communicate well with most people yet.

Actually, Falvey was probably not too far off with respect to the available mass-market computer technology of the times. The *Wall Street Journal* on that day also contained a full-page ad for Compaq, which lauded the size and power of the Compaq Portable Computer: 128K bytes of RAM, expandable to 640K; one 320K minifloppy disk drive, optional second; and a size of 20" X 8.5" X 16"– "designed to fit under a standard airline seat."

The *Los Angeles Times*' contribution was a Berry's World cartoon depicting a tot sitting before his computer, with Dad looking quizically over his shoulder at the screen. The tot was saying to his friend beside him "Don'tcha LOVE to get adults on your own turf– where THEY feel intimidated?" [6]

However, the popular computer press of the day, was much more attuned with the times. The August 1983 issue of *BYTE* contained a perceptive article written by Bill Gates entitled "The Future of Software Design" [7]. Gates identified five major issues facing software developers in 1983.

1. Integration: building software packages that work together.
2. User Interface: presenting data on the screen, incorporating graphical icons and windows into standard user interfaces.
3. Data-Storage Metaphors: simplifying the ways in which users perceive data.
4. Tying Personal Computers to Mainframes: tying personal computers to mainframes in a way that would allow automatic database querying.
5. Expanded Definition of an Operating System: incorporating increasing number of functions (e.g., graphics capabilities, user-interface capabilities, networking) within the operating system.

Gates' conclusion: "The revolution is here – and it is soft." Following Gates' paper was an article entitled "The 8086–An Architecture for the Future."

In 1983, hardly anyone (other than those involved in the Anderson study [3]) could conceive of a computer user (presumably authorized) being "malicious." The idea that computers might themselves be malicious or that their software might be capable of contracting and spreading "viruses" was ludicrous. Thus, we should not be surprised to see that the majority of the requirements contained in the TCSEC (i.e., those for classes B1 and below) are designed to protect against "bad guys" getting into the system from the outside, but assume that authorized users are basically "good guys" and that the software they use is safe. Only the highest three classes of the TCSEC (B2 through A1) contain assurance requirements designed to protect against the malicious user envisioned by the Anderson panel ten years before the TCSEC was published:

> Until now, the principal threat has been seen to be an external penetration. The primary defense against external penetration has been that of preventing access to any part of the system or its data. The malicious user concept on the other hand has bypassed this form of defense by assuming that the malicious user has legitimate access to a system. Taken in the context of open use systems with general programming available to all users, it is clear that the defense against a malicious user must reside in the process that controls the operation and execution of arbitrary programs. [3]

Note that even the Anderson panel did not envision malicious "authorized" software capable of acting independently of a user, as we see in "worm" and "virus" attacks. So, within this historical context, is it any wonder that vendors over the past 10 years have focused on building low-assurance products? Their market – including the Department of Defense – has failed to recognize the need for assured protection against the insider threat. Only a few vendors (primarily those whose business is security) have ventured into the challenging and risky area of attempting to build truly trustworthy systems. Thus, most of our "trusted systems" provide security-related services, but very little assurance that these services operate as presumed and cannot be by-passed or subverted.

2 The Emerging Malicious User

The world has changed considerably since the TCSEC was written; it has become much smaller. We now live in a world community with a global economy and global politics. We send electronic mail to the other side of the world as easily as to the next office. Many of our tried-and-true "bad guys" have become "good guys," some of our "good guys" have become "bad guys," and we have a few "two-hatted" world neighbors.

Computer technology since the TCSEC was written and since the forward-thinking Anderson study was conducted has also changed enormously. The concept of the central processor with multiple user accounts sharing resources has been replaced by distributed systems, multimedia, mega-databases, client-server architectures, pandemic networks, and of course "Macs." Gates' software revolution is here in full force, and the issues he addressed have become major design drivers for current computer technology. The predictive powers of the 13-year-old student in the Bronx clearly have dramatically surpassed those of Falvey! Computers have really "come into the world" (or possibly vice-versa).

This 13-year-old kid is now 23. Let's suppose he opted for a military career. Having been selected for the summer program in the Bronx, he is probably a star lieutenant. So he possibly is the operator of a highly-critical Department of Defense system. As with most critical applications, the most important requirement is that the mission be carried out when necessary – which may not be too often, in reality. Also, as with most critical applications that require inputs from multiple data sources, the system is connected with other systems throughout the world. So what do you suppose this young, bright, and bored lieutenant, who learned to use the computer when he was in computer camp ten years ago, is going to do with his time? He may even bring in his favorite game program from home to occupy his time. Voila! The malicious user emerges!

3 Does the TCSEC Paradigm Fit?

I contend that the most serious threat to computer systems today is the computer literacy – and "computer friendliness" – of users with a naive trust in computer technology. Whereas in the past, we have been able to rely upon user naivete and technology intimidation to reduce risks associated with known and unknown vulnerabilities, I do not believe we can afford to do so today. The growing plethora of software programs available from large software houses, small home-based businesses, and "on the street" exacerbates the problem.

Truly trustworthy systems (i.e., those needed for our most critical applications) must effectively protect against both malicious and "adventurous" users, and against misbehaving programs. They must be easy to use safely; adding (or upgrading) applications and ensuring that they will run securely should be no more difficult than just adding and running them. We must be able to build trustworthy networks, multimedia applications, distributed systems, networked personal computers, and client-server architectures, as well as central, shared systems. We must be able to build systems that remain trustworthy despite deviant and misbehaving code, and despite the actions of malicious, mischievous, curious, and careless users.

Are the functional and assurance requirements the TCSEC defines sufficient to counter this threat environment? Clearly, the functional requirements designed to protect a central processor shared among multiple users cannot accommodate the myriad of existing and evolving software and hardware configurations. Attach an evaluated system to a network, and "all bets are off;" write an application that uses privilege, and "all bets are off"[1]; compose a system of evaluated products, and it's anyone's guess what level of trustworthiness you have. The new Federal Criteria currently in development is attempting to remedy this problem by separating functionality from assurance, and the current draft contains requirements to facilitate safe use, another area lacking in the TCSEC [8]. But the bottom line is that TCSEC functionality is not adequate for securing current processing environments against the existing threat.

Let us look at the TCSEC assurance requirements; i.e., those aimed at providing confidence that the system is as trustworthy as it purports to be. Are the assurance requirements for classes C2-B1 sufficient for this threat environment? Most definitely NOT; nor were they ever intended to be. Classes C2-B1 primarily provide security services to cooperating, non-malicious users; these classes are not designed to counter malicious threats or the effects of innocent errors. Can the assurance requirements for the more highly rated systems (B2 through A1) provide sufficient trustworthiness to counter this threat? I believe that answer is YES. The security engineering that goes into ensuring that a system meets the B2 modularity requirement, or the stringent B3-A1 requirements for well-defined layering and minimization, provide significantly greater assurance in the system's correctness than is possible from a C2 or B1 system. The use of formal methods at class B3 and especially at A1 adds further assurance of correctness. The stringent evaluation and penetration testing required for a B2 or higher rating provides significantly greater assurance in the system's trustworthiness. In fact, the only noteworthy developments in computer system assurance methods made in the past 10 years are object-oriented design (which can

[1]In fact, even EMACS (a sophisticated text editor), which does not require privilege to run but runs with the user's accesses, makes discretionary access controls behave in unexpected ways.

be considered a variant of the TCSEC's system architecture requirements) and Computer Aided System Engineering (CASE) tools (which also can contribute to the TCSEC's system architecture requirements). Unfortunately, designing and building highly assured systems is both time-consuming and expensive – both of which are anathema to the current technological and economic environments.

Is it possible to identify broad functional requirements responsive to the rapidly evolving technology and threat environment? Is it possible to develop "trusted" systems that are truly "trustworthy?" Is it possible for "trustworthy" systems to be user-friendly while meeting performance requirements? The assurance requirements identified in the TCSEC are aimed toward building a sound foundation upon which to build functionality – a high-integrity architecture that can withstand assaults from malicious and misbehaving users and application code. These assurance requirements are as applicable today as they were when the Anderson panel made its recommendations and when the TCSEC was written. Unfortunately, very few computer systems meet these requirements; in fact, the trend is away from assurance toward increased functionality and complexity. Very few system developers even make an earnest attempt – largely because the evolved threat has not yet been acknowledged by their markets enough to make building trustworthy systems economically viable.

To create a demand for security assurance requires that the threat environment be realistically characterized. Both malicious and non-malicious threats must be considered, as well as the constraints inherent in the operational environment (e.g., response times, operations concept, stability). Further, the characterization of a threat environment cannot be based upon the asserted trustworthiness (i.e., clearance, absence of malicious intent) of users or on the presumed innocence of application software.

The TCSEC legacy clearly is inadequate to counter the evolved threat. Its functionality requirements are too rigid and technologically archaic, and it does not adequately address malicious execution. Even its covert channel bandwidth guidelines appear antiquated in light of current processing power. However, its assurance requirements remain valid – and more important than ever.

4 The New Paradigm

What is needed to counter the new threat paradigm is a highly assured "security infrastructure" that will be responsive to rapidly changing technology and demands for increasing functionality and assurance. Such an infrastructure would provide a framework within which computer system technology could expand and grow without sacrificing security assurance. This infrastructure must address the same issues Gates identified ten years ago. However, notably none of these issues directly relates to security functionality or assurance. In fact, because the technical solutions for these issues to date have significantly added to the complexity of computer systems, they actually have made the challenge of building trustworthy systems – and a trustworthy infrastructure – more difficult! The security infrastructure must address these issues in at least the following ways.

1. Integration. The infrastructure must include trusted products that can work together to achieve a consistent level of trustworthiness; standards for labeling schemes that support open systems; and rules governing composibility to achieve desired levels of protection.

2. User Interface. The infrastructure must include multilevel windowing systems; multilevel graphical presentations; and application interfaces that are easy to use without having to trust tens of thousands of code statements.

3. Data-Storage Metaphors. The infrastructure must include multilevel database management schemes that facilitate the realistic classification of output derived from data with multiple sensitivity levels.

4. Tying Personal Computers to Mainframes. Although the era of the mainframe appears to be waning, the infrastructure must include the capability to tie single-level PCs and workstations together as multilevel networks; security mechanisms for tying single-level systems into multilevel servers; the ability to securely tie trusted systems together; and the ability to build trustworthy networks, multimedia applications, distributed systems, networked personal computers, and client-server architectures.

5. Expanded Definition of an Operating System. Gates' goal of "incorporating an increasing number of functions within the operating system" is

highly *undesirable* in the security infrastructure because it increases the complexity of the portion of the system that must be engineered, proven, and maintained correctly in order to assure the system's trustworthiness. As more functionality is added at lower levels of a system (e.g., expanding functionality within the operating system), the system becomes more complex, and security assurance becomes less tractable. So the security infrastructure must attempt to *minimize* the functionality that must be engineered correctly, proven correct, and maintained securely. To gain required levels of assurance, the security infrastructure must include operating systems that take advantage of multi-state hardware to enhance both protection and performance; security policies that address data/program integrity, service-assurance, program sensitivity, and safety; capabilities to protect against misbehaving programs[2]; and new methods for engineering trust into systems, for proving and quantifying assurance, and for maintaining assurance through the system life cycle despite deviant and misbehaving code, and despite the actions of malicious, mischievous, curious, and careless users.

Most importantly, the security infrastructure must by accompanied by social change: recognition that "good guys" sometimes wear "black hats." Until this recognition comes about, computer vendors will continue to market and sell security services with little assurance that they work as advertised or are effective in countering the targeted threat. Undoubtedly society will come to realize that both authorized users and software are capable of committing malicious and highly destructive acts. Unfortunately, this social change is already behind the threat curve.

References

[1] *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, Department of Defense, Washington, DC, December, 1985.

[2] *Trusted Computer System Evaluation Criteria*, CSC-STD-001-83, National Computer Security Center, 15 August 1983.

[3] Anderson, J. P. *Computer Security Technology Planning Study*, ESD-TR-73-51, Vol. 1, Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, MA, October, 1972.

[4] "Computers Draw 1,100 To Classes." *New York Times*, August 15, 1983, p B3.

[5] Falvey, J. "Don't Count Too Heavily on that Personal Computer." Manager's Journal, *The Wall Street Journal*, August 15, 1983, p 16.

[6] Berry's World. *Los Angeles Times*, August 15, 1983.

[7] Gates, W. "The Future of Software Design." *BYTE*, August 1983, pp. 401-403.

[8] *Federal Criteria for Information Technology Security* (DRAFT). National Institute of Standards and Technology and National Security Agency, December 1992.

[2] Including malicious programs that exploit covert channels – the multilevel-generation's "epidemic."