Editor: O. Sami Saydjari, ssaydjari@cyberdefenseagency.com

Multilevel Security: Reprise

O. Sami Saydjari
Cyber Defense Agency

For the first three decades in the field of computer security (1960s, '70s, and '80s), everyone focused on multilevel security. Why did the focus all but disappear in the '90s? Was there a major failure in multilevel security? Is there less need for multilevel security today? If there is still a need, how might we address it?

To define multilevel security, we must first define security, which is "the combination of confidentiality (the prevention of the unauthorized disclosure of information), integrity (the prevention of the unauthorized amendment or deletion of information), and availability (the prevention of the unauthorized withholding of information)."[1]

More generally, security is freedom from risk. An information system's intrinsic value is dwarfed by that which it supports: an organization's mission. Security, then, is the freedom from risk to a mission.

"Multilevel" refers to data classes that are treated differently depending on which users access them.[2] The word "level" implies a class hierarchy, such as top secret, secret, confidential, and unclassified. More commonly, however, it refers to the general concept of access-class differences, which don't require a strict hierarchy.

Multilevel security is the prevention of unauthorized disclosure among multiple information classes. The threat source for the disclosures includes unauthorized users and subverted software operating on behalf of authorized users.[3] The terminology might be more explicit if we could call this concept multidomain confidentiality, but it is worth resisting multiplying terminology. Nonetheless, we should understand that multilevel security means multidomain confidentiality.

It's important to understand what multilevel security is not. It is not complete because it doesn't address integrity and availability. What's more, it does not imply anything about the assurance level—referring only to a system's functional ability to distinguish among data classes, user classes, and access rules between the user and data classes.

The need

Why not have separate computers for each data class, one for confidential, one for secret, and so on? In computer security's early days (1960s and '70s), computers often weren't interconnected and were prohibitively expensive by today's standards. Was the need to handle multiple data classes on the same computer a cost-savings issue—to avoid buying separate computers for each data class? It wasn't, yet many information assurance professionals still believe that myth today.

The real multilevel-security driver was the imperative to integrate different data classes to accomplish a mission. For example, to develop a secret military operations plan, you must consider intelligence information usually maintained at the top-secret level, and then create primarily unclassified subplans to logistically support the operation. Similarly, top-secret intelligence reporting needs to employ up-to-date secret and unclassified information. In other words, military missions are inherently multilevel.

Is the military the only entity that needs multilevel security? Certainly, the commercial world must protect multiple-class data, but its motivation is for commercial business purposes[4] and includes limiting data-class accessibility according to an employee's job classification. For example, engineers shouldn't be able to access personnel records, and accountants shouldn't be able to access proprietary engineering designs. Furthermore, commercial processes require integrating various data classes. But merely performing a bid and executing a contract requires access to personnel pay information, accounting systems, and engineering designs. Outputs from each of these systems must flow to the others. Thus, the business world's mission is also inherently multilevel.

Military and commercial requirements for multilevel security are shared, but each has a different emphasis. The major difference is in the degree of assurance required. The threat to military systems is greater in the sense that adversaries are willing to invest significantly more to compromise a military system than a commercial one. Fur-

thermore, commercial organizations can sometimes insure their residual risks. It is difficult to insure residual risk to national sovereignty.

The approach

In the 1960s and early '70s, when the requirement for multilevel security was first articulated, it wasn't clear that anyone knew how to build systems that truly met such requirements. The US government invested hundreds of millions of dollars in a series of research programs to discover and demonstrate how to create trustworthy multilevel systems, including Multics Access Isolation Mechanism (Multics AIM), Provably Secure Operating System (PSOS), Kernelized Virtual Machine (KVM), Kernelized Secure Operating System (KSOS), Blacker, Secure Ada Target (SAT)/Logical Coprocessing Kernel (LOCK), Trusted Mach (TMach), and Distributed Trusted Mach (DTMach). Each research prototype took more than five years to create.

The theory behind several of the research efforts was to reuse the code base across multiple projects, allowing industry to use government prototypes to create mainstream products. The reality was that the commercial products had a small customer base that numbered only in the dozens, and the systems took so long to create that their technology base changed significantly before they could be marketed. At that time, the computing paradigm was shifting from stand-alone time-shared mainframes to personal-computer (PC) networks whose members were nearly as powerful as the old mainframes.

Meanwhile, the US Department of Defense (DoD) published the Trusted Computer System Evaluation Criteria (TCSED)[5] and set up a program to assess system compliance with the criteria's six increasing trust levels (C1, discretionary security protection; C2, controlled access protection; B1, labeled security protection; B2, structured protection; B3, security domains; and A1, verified protection). The evaluation program's primary goal was to assess commercial products. The program's theory was that the government demands for trustworthy multilevel security would be significant enough to attract commercial companies to build security into their main product lines. Unfortunately, most evaluations occurred at lower assurance levels, meaning that few systems targeted the levels that were the primary motivation for the standards in the first place.

The first evaluations took a substantial amount of government and commercial time and resources. Worse, vendors didn't know how much time the evaluation process would take, which affected time-to-market, a key factor to a product's success. Vendors thus became disenchanted.

In parallel with the evaluation process, the US National Computer Security Center, in cooperation with the US DoD, developed a policy that required DoD procurement to specify systems that met TCSEC standards. Concerns over long development times, evaluation-time uncertainties, and application compatibility issues began to shake the resolve of DoD programs that initially intended to comply with the policy to acquire systems evaluated against the new standard.

Slowly, DoD programs began waiving their multilevel security requirements; exceptions became the rule and, ultimately, the policy completely eroded. Also, as the computer market exploded, the DoD went from being the major computer consumer to a minor market share. Together, these factors caused the financial incentive for commercial development of multilevel systems to all but disappear.

The confluence of these circumstances left a bad taste in government sponsors' mouths: multilevel security became a bad word and funding dried up.

The trend

While the situation I just described was unfolding, computers were becoming more pervasive in the user community. Organizations increasingly depended on them to accomplish their missions.

Computers raised organizational operations' tempos and opened doors to integrating activities. Linking a producer's computers with those of its suppliers and distributors improved efficiency. Developers increasingly squeezed out system redundancies to achieve increased efficiency. With a decreasing need for large inventories, organizations moved to just-in-time models. These changes dramatically increased productivity, but they also increased system brittleness—failure probabilities and their consequences. The military experienced the same trends. Thus, processes that once were inherently multilevel remained so.

Those who thought that lots of cheap computers would solve the multilevel security problem fundamentally misunderstood the problem.

As computers became increasingly interconnected and central to mission functions in the 1990s, requirements amplified because criti-

cal cross-level processes increased, and their span began to cross organizational boundaries. Those who thought that lots of cheap computers would solve the multilevel security problem fundamentally misunderstood the problem.

We must balance risk against functionality and performance—optimize a mission's success probability, not just minimize system risk.

Enter the guards

Given that the need for multilevel secure systems was rising and the supply was falling, how could the demand be met? Enter the *guards* (and their lower-assurance cousins, gateways and firewalls). One way to achieve inherently multilevel processes on single-level systems is to interconnect multiple single-level systems via guards. A guard is a "processor that provides a filter between two disparate systems operating at different security levels or between a user terminal and a database to filter out data that the user is not authorized to access."[6] The guard allows a type of mirroring of a priori-specified data subsets from one single-level system to another.

The problem is that once data classes mix, we must treat all the data as the same class because a single-level system can't assuredly maintain data class separation (for example, separating secret from unclassified data). Of course, the whole point of operating a single-level system (such as computer networks) is to avoid needing assured multilevel computers, so assured data class separation isn't possible.

To share resulting data products with a different class, we must determine which parts to share. We must be able to confidently label the designated parts, and the system must assuredly use those labels to push only the sharable parts to the other class. Humans must perform the designation; the program that creates the sharable product must be demonstrably functionally correct. This makes the process labor intensive, error prone, and risky.

So, the multilevel security problem didn't get solved; we handed it off to guards. Guards become high-profile targets because they're the adversaries' gateways to large quantities of important data. Guards are also the worst place to make data-labeling decisions because they're distant from the data's originators. The risk for these solutions is considerable, and it continues to grow as we interconnect more systems and increasingly rely on computer support for mission-critical functions.

Weighing risk

Security, like many other goals, is a matter of risk versus benefit. How much is multilevel security worth? Consumers appear to avoid pervasively deploying available multilevel secure systems because of expense, a perceived compatibility problem with their application programs, and potential functionality loss because security policies restrict some activities. Consequently, users operate at single-level security most of the time.

When inherently multilevel secure tasks occur at a single level, there must be a trade-off. Perhaps decision makers choose incorrectly because they fail to use information that might have been useful in preparing a report. Or a financial transaction fails because some information wasn't disseminated to all involved parties. It's hard to measure all the consequences of an information-sharing failure because, given the potential results' seriousness, no one wants to experiment.

Risk management versus risk avoidance

For some highly sensitive systems, designers used to try to prevent any attack they could imagine, independent of an attack's success probability. Similarly, the DoD accreditors often refused to authorize functions and interconnections if there was any measurable residual risk. Within the past 10 years, as systems have become more complex and we've begun relying on them for critical functions, we have been forced to balance risk against functionality and performance—optimizing a mission's success probability, not just minimizing system risk. This is an important paradigm shift.

Because we have no good means to measure the risks or benefits that contribute to mission-success optimization, it isn't clear how to explicitly and thoughtfully make this trade-off. What we need is a good engineering metric for risk and the tools and techniques to trade off risks against benefits in terms of mission optimization.

Minimizing costs

To decide how best to invest to solve the multilevel security problem, we must look at what a system lifecycle costs. Unfortunately, few organizations measure costs this way. For example, in the development phase, programmers must deliver results at minimum time and cost. Organizations don't factor in the cost of a potential security breach that results from a poor design decision. Yet, if we are to make

reasoned investment decisions, that is exactly what we must consider.

Train as you fight and fight as you train

Imagine a battlefield exercise in which one team's new experimental tanks has cardboard "armor." Then imagine the opposing team receives tank-fighting instructions: don't aim at the tank, use real bullets, or even use rifles; just throw the blanks really hard—and only if the tanks aren't doing something operationally important because we don't want you to upset the exercise.

This is how most military and business exercises use red teams, which often do serious damage despite operating under such severe limitations. When deployed information infrastructure systems fail during a red-team attack, the defense force commander and the program's executive office (which built the weak system) are accountable; someone must take remedial action. Risk–benefit trade-offs reach productive levels only when there is a feedback loop between risks and undesirable outcomes. Risk takers must be able to adjust risk levels. Break the feedback loop and risk-takers might take unacceptably high risks because they don't pay any penalties.

Multilevel security is a linchpin problem because without it, organizations simply can't perform their missions. The difference is only in the degree of multilevel sharing that's enabled and the risk level an organization assumes. Existing technology solutions (for example, Gemini's Gemsos kernel, Trusted Computer Solutions' Trusted Gateway, and DigitalNet's XTS-300) facilitate more integrated sharing and potentially decrease risk. Several proposed architectures using those technologies look promising.[7] But we don't understand their functional impact, the total implementation costs over a system's life cycle, or the actual risk reductions we gain. It's little wonder that we're confused on how to balance risks and benefits.

Similarly, R&D can provide many technological improvements to multilevel security technology. Multilevel systems should support the full functional richness of user applications, facilitate cross-domain sharing, reduce covert channels, include improved integrity mechanisms, and so on. Yet, until the security community understands how to measure mission-optimization risks and benefits, it's difficult to make a good case for either increasing deployment of existing solutions or improving technologies.

Developing risk and benefits metrics is not magic; it's hard work. But given the increasing risks the operational community takes each day, and their confusion over the appropriate course of action (due, in part, to a lack of metrics), it's an investment that's overdue. The payoff would immediately begin addressing the multilevel security problem. The metric would drive operational solutions today and guide research toward better solutions.

Multilevel security is necessary to realize the military and commercial visions for tightly integrated systems. Thus, we must embrace the multilevel security problem with renewed vigor. We must precisely define what it means to solve the multilevel security problem from the operational mission perspective. Well-defined use cases representing the desired functionality from a variety of application areas would be illuminating. We must define metrics for the solutions to judge their effectiveness. We must define what types of integration and functionality to avoid because they pose too much risk. We must define what key theory issues remain unsolved (for example, the covert-channel problem) and articulate those problems precisely so that the research community can address them. We must also define experimental computer-science methodologies to scientifically apply the emerging metrics to proposed solutions to assess their true effectiveness. With these intellectual tools in hand, we might begin the journey with confidence down the challenging road of addressing the multilevel security problem. □

References

1. *Information Technology Security Evaluation Criteria (ITSEC)*, Commission European Communities, 1991.
2. *Information Security: An Integrated Collection of Essays*, M.D. Abrams, S. Jajodia, and H.J. Podell, eds., IEEE CS Press, 1995, pp. 40–59.
3. E.A. Anderson, C.E. Irvine, and R.R. Schell, "Subversion as a Threat in Information Warfare," *J. Information Warfare*, vol. 3, no. 2; www.jinfowar.com/extranet/article.asp?section=extranet&ContentID=jiw32_5.
4. D.D. Clark and D.R. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proc. IEEE Symp. Computer Security and Privacy*, IEEE CS Press, 1987, pp. 184–194.
5. *National Computer Security Center, Department of Defense Trusted Computer Security Evaluation Criteria, DOD 5200.28-STD*, 1985.
6. *Instruction No. 4009, Nat'l Information Assurance Glossary,* Committee on National Security Systems (CNSS), 2003; www.nstissc.gov/assets/pdf/4009.pdf.
7. N.E. Proctor and P.G. Neumann, "Architectural Implications of Covert Channels," *Proc. 15th Nat'l Computer Security Conf.*, 1992, pp. 28–43.

***O. Sami Saydjari** is CEO of Cyber Defense Agency, LLC, an information-assurance research and development lab and consulting firm. He is also the president of the Professionals for Cyber Defense, a nonprofit group of information-assurance leaders dedicated to helping the US government adopt sound cyberdefense policy. Contact him at ssaydjari@cyberdefenseagency.com.*