Fortresses Built Upon Sand

Dixie B. Baker, Ph.D.
Science Applications International Corporation

The current "trusted system" paradigm is built upon the notion of a Reference Monitor that assumes the existence of a well-defined security policy, a bounded system entity, and a centralized reference validation mechanism with knowledge of and control over the system entity. The "trusted system" paradigm is hierarchical: management defines the policy, the hardware and system software that comprise the trusted computing base enforce the policy, and applications must conform to the policy. This paradigm acknowledges that applications depend upon the hardware and operating system on which they run, and that *assurance that they will execute safely is derived from the strength of this "trusted computing base."*

Several observations have prompted computer scientists to reexamine and question the relevance of this hierarchical "trusted system" paradigm:

- Individuals, businesses, government, and social services are increasing their dependence upon computer systems and networks for both routine and critical functions, including electronic commerce, communications, education, medical collaboration, and entertainment.
- The information systems upon which society is becoming increasingly dependent (e.g., power grids, telephone, Internet) are highly complex and non-hierarchical, often lacking a clear boundary and a common set of security objectives.[1]
- Attacks on networks and computer systems are becoming more frequent, virulent, global, and broadly publicized.

The "obvious" conclusion seems to be "The Emperor has no clothes!" The "trusted system" paradigm must not be working – what we need is a totally different paradigm!

It's obvious to even the casual observer that what we're doing now to make our systems safe and secure is not working. But is the "trusted system" paradigm at fault? Or are we just attempting to build our fortresses upon sand? Let's examine the perceived problems with the existing paradigm and one of the proposed solutions.

Policy

In his provocative paper "The Emperor's Old Armor" [BLA96] (different Emperor), Bob Blakley argues that "policy" is too complex (difficult to administer) and is not scaleable. My first reaction to this assertion was "without a policy, how do you know what you're trying to

[1] I am consciously avoiding using the term "security policy," as it appears to have different meanings for different people.

accomplish?" Then I realized that Blakley was really talking about access control policy and not "policy" in general. In that case, I agree with him that access control policies are very difficult to administer, and don't scale in any dimension; they are limited both in the number of subjects, functions, objects, operations, and classes, and in breadth of control. Further, as both Wulf et al.[WUL96] and Greenwald [GRE96] point out, today's distributed environments require multiple protection philosophies and policies.[2]

Though not articulated, the common "policy," or perhaps more accurately "philosophy of protection," that runs throughout all of these papers (and others that were presented at the New Security Paradigms Workshop) is that systems should be *dependable.*[3] Basically, all of the approaches presented expect, assume, and depend upon systems to:

- Behave predictably; they should do what we think they will do.
- Be available when we need them.
- Be safe; they should not do what we don't want them to do.
- Be capable of protecting our data from unwanted disclosure, modification, and destruction.
- Respond quickly.

In other words, systems should be *trustworthy.*

Greenwald presented an interesting and useful multi-policy approach for addressing the challenges of resource management and access control across distributed systems policies. But, as he acknowledges, a user would first need to identify and authenticate herself to a particular host in the distributed system before the security policy could be enforced by the Distributed Compartment Model (DCM) mechanisms. Thus the DCM depends upon the underlying operating system to provide fundamental protection, and the DCM mechanisms add a layer of more finely grained access control. Similarly, Wulf et al. acknowledge that Legion comprises high-level mechanisms and objects that must trust the host machines on which they reside and that the project has not addressed lower-level mechanisms such as messaging.

System Integrity and the Reference Monitor Concept

Blakley's argument appears to assume that existing systems actually use trusted systems that implement the Reference Monitor Concept. Not so! In fact, the vast majority of computer systems in use today are first-generation systems originally designed for use in the home. Unfortunately, these "personal computer" systems have migrated into the business environment, where none of the assumptions upon which they were built hold. They are physically unprotected, accessed by multiple users, and used for critical business functions. They are neither predictable nor safe (though they are all too available.)

---

[2] H. Hosmer introduced this concept at the 1992 New Security Paradigms Workshop [HOS92].

[3] "Dependability" is a broad term defined as the degree of trust that may justifiably be placed in a system's reliability, availability, safety, security, and performance [CRI95].

But unfortunately, as the PC has become ubiquitous, a degree of complacency and an acceptance of the inevitability of software failures have emerged along with the technology. People have come to expect program failures. As contrasted with the guarantees and warnings that are expected to accompany physical products ranging from hair dryers to baby strollers, commercial software invariably carries sweeping disclaimers that in effect say "If this program doesn't work, tough luck!" In his book *Fatal Defect* [PET95], Ivars Peterson challenges his readers to "Imagine your reaction if you found [such a] disclaimer on your car or kitchen appliance... [and] ... what would happen to our society if everybody who wished to use a telephone, television set, car, detergent, or plastic toy were first obliged to learn at least a little about how it was made and how it works internally, and then to test it for hazards and other surprises." Peterson poses the rhetorical question "Why are software manufacturers allowed a sweeping disclaimer that no other manufacturer would dare to make?"

I doubt anyone would argue with Blakley's assertion that neither perfection nor correctness is a reachable goal in computer systems – nor that "system integrity is hard." But we appear to be going in the wrong direction. Instead of moving computer science forward in developing dependable systems, we appear to be going backwards in our acceptance of mediocrity. "Correctness" is not the issue; "dependability" is.

All of the papers presented in this session[4] depend upon the underlying operating system to carry out the actions of the applications – whether they be "Legions," "handles," or "agents." Dependencies in computer systems cannot be ignored. No computer system component can be any more trustworthy than the components upon which it depends. So while the trusted computing base may be the weakest link, it's also the most essential.

Applications derive their functionality and assurance from the strength of the underlying infrastructure; unfortunately, the dependability of that infrastructure has largely been ignored.

Emergent Security

Concepts relating to complex adaptive or "ecologic" systems (e.g., genetic algorithms, software "agents," "emergent" behavior, adaptation) appear quite appealing for building an alternative approach to securing very-large-scale systems, such as the Internet. In complex adaptive systems, simple behaviors of individual agents create powerful group behaviors. So why not create software "agents" capable of assimilating their environment and "learning" adaptive, protective behaviors that collectively will result in the emergence of a secure system? Such an approach is highly attractive in that it appears to mimic how complex systems develop and to eliminate a "brittle" dependency.

Rasmusson and Jansson [RAS96] (as well as others) have suggested approaches based upon the ecologic model. Rasmusson and Jansson suggest using commerce agents that detect and avoid

[4] "Best of the New Security Paradigms Workshop," National Information System Security Conference, Baltimore, MD, September, 1996.

interaction with malicious participants, causing "reputations" and secure electronic commerce to emerge. However, security approaches based upon adaptive "agents" must deal with three fundamental problems:

1. Values are not universally shared.
2. Time is not an ally.
3. The systems upon which the agents rely are not necessarily trustworthy (if they were, the agents might be superfluous).

Values are not universally shared.

"What's best for the community" in the "new" paradigm is equivalent to a "security policy" in the "old" paradigm. Experience has shown that agreeing upon a security policy even within a single enterprise is an arduous task; and as Greenwald and Wulf et al. have observed, a single security policy may not be appropriate in a distributed or complex system.

As an example of how social control evolves, Rasmusson and Jansson contrast what happens in a small village with what happens in a large city. In a small village, life is governed by the social control that evolves from the individual values of the residents. In a large city, the large number of people increase the variability of values, and reduces social control. Complex computer systems such as the Internet are much more similar to a large city than to a small village, so the probability that "safe" social values will naturally emerge is much lower than for a single business enterprise.

So how does one assure adaptation to the "right" policy or set of policies? What happens if the agents "learn" the "hacker's ethic?" Would that be acceptable to everyone? How does any single participant in the community know what he or she is being protected from? Is it important to know? Who decides who decides?

Time is not an ally.

Adaptation takes time. Experience has shown that attacks can be accomplished incredibly fast. Consider the Morris Internet "worm." Late in the evening of November 2, 1988, Robert Morris, Jr., released into the ARPANET a "worm" program that expropriated the resources of each invaded computer and generated replicas of itself on other computers. Within hours, it spread to several thousand computers attached to the worldwide research network [DEN90]. Could agents, without knowing the precise nature of the impending attack, have adapted quickly enough to protect the ARPANET "community" from this virulent attack?

The systems upon which the agents rely are not necessarily trustworthy.

Most importantly, software "agents," like any other applications, are dependent upon the strength of the infrastructure on which they execute. Since software agents are nothing more than application programs, this means they themselves are vulnerable to attack and subversion by other applications (which may be other agents). Indeed, a virus can be viewed as a software agent in that it causes executable programs to "learn" whatever behavior the virus is transmitting. As Blakley has noted,

> Intelligent Agent" architectures invite us to execute other peoples' code on our systems and to write our own code and send it out to make its way in the world without benefit of our oversight. These agents are not distinguishable from programs we used to call "viruses."

While some argue that not all viruses are "bad" (see "Values" discussion above) and that they can perform useful functions, the typical virus does not perform functions the majority of people desire their systems to "learn."

Security has historically been viewed as a "weakest link" attribute. In fact, the "brittleness" of the "trusted system" paradigm relates to the single-point-failure problem. *Any system component is only as trustworthy as the components upon which it depends.* The validation mechanism provides both the strength of the security architecture and its primary point of vulnerability.

The Bottom Line

No matter how one may try to deny it, ignore it, or distort it, the simple fact remains; security is built from the foundation up, or to quote Blakley, "security must be inherent, not imposed." None of the "new paradigms" presented at the 1996 New Security Paradigms Workshop can overcome the simple fact that very few of the operating systems and networks in use today are dependable or trustworthy. The "trusted system" paradigm is as relevant as it ever was; it just was never taken seriously. To assure that applications can execute safely – whether they be on a home computer, in a distributed business system, or on the Internet, and whether they take the form of "Legion," "handle," "agent," or a critical healthcare application – requires a dependable infrastructure, which is what "trusted systems" attempt to achieve. Unfortunately, few consumers have demanded that to this point.

As Blakley points out in a footnote:

> The temptation to condemn vendors as lazy or irresponsible for underspending on assurance is powerful; however, the security community has not made a very good case that the market will support the cost of assurance – or even that money for assurance is well spent.

What will it take?

A Word about Secrecy

Blakley also argues that depending heavily on secrecy for security will not work because people can't keep secrets. I'm in total agreement here. In fact, I would add that a direct correlation appears to exist between degree of secrecy and the amount of effort people are willing to put into getting to it (and the more pain they're willing to endure). I think an interesting experiment would be to offer two talks at a conference: one advertised as a classified talk on "Verb

Conjugation"[5] and the other advertised as an unclassified talk on "Microsoft's Latest Idea." I have a feeling we would find far more people interested in verbs than we might have guessed. The best way to protect secrets might be to post them on billboards.

References

[BLA96] Blakley, B., "The Emperor's Old Armor," to be published in *Proceedings of the New Security Paradigms Workshop*, 1996.

[CRI95] Cristian, F, G. Le Lann, and T. Lunt (eds.), "Dependable Computing for Critical Applications 4," ISBN 3-211-82649-1, *Dependable Computing and Fault-Tolerant Systems*, Volume 9, 1995.

[CRO95] Crosbie, M., and E. Spafford, "Defending a Computer System using Autonomous Agents," *Proceedings of the 18th National Information Systems Security Conference*, Oct 1995.

[DEN90] Denning, P. J., "The Internet Worm," *Computers Under Attack: Intruders, Worms, and Viruses*, Peter J. Denning, Ed., ACM Press, 1990, Chapter 10.

[GRE96] Greenwald, S., "A New Security Policy for Distributed Resource Management and Access Control," to be published in *Proceedings of the New Security Paradigms Workshop*, 1996.

[HOS92] Hosmer, H. H. "The Multipolicy Paradigm for Trusted Systems," *Proceedings of the New Security Paradigms Workshop, 1992-1993*, IEEE Computer Society Press, Los Alamitos, CA.

[PET95] Peterson, Ivars. *Fatal Defect: Chasing Killer Computer Bugs*, Times Books, 1995.

[RAS96] Rasmusson, L. and S. Jansson, "Simulated Social control for Secure Internet Commerce," to be published in *Proceedings of the New Security Paradigms Workshop*, 1996.

[WUL96] Wulf, W. A., C. Wang, and D. Kienzle, "A New Model of Security for Distributed Systems," to be published in *Proceedings of the New Security Paradigms Workshop*, 1996.

[5] Apologies to the grammar buffs.