Integrating Formalism and Pragmatism: Architectural Security

Ruth Nelson
Information System Security
48 Hardy Ave.
Watertown, MA 02172

Abstract

Two major schools have dominated computer security research and thinking for the last twenty to twenty-five years - formalism and pragmatism. In spite of all their work, progress has been limited, system designers do not incorporate security principles and security is seen as a detriment to functionality. This paper suggests that an architectural approach, fostered by better communication among security researchers and between the security and operational communities, may yield more practical and effective security solutions. These architectural solutions are not totally general, but they have a structure and are applicable to large classes of problems.

Introduction

Two major schools have dominated computer security research and thinking for the last twenty to twenty-five years. One of these, which I will call the formalist school, focuses its work on correctness and on universality. They have developed abstractions of security properties, notably access control, which are intended to apply in a system independent manner. Assurance in a computer system means that the system is designed to preserve these security properties, and that the design has been proved to do so, using a variety of formal methods.

The other school, the pragmatists, are concerned with attacks on and countermeasures for real systems. The best known efforts are the research and development work in intrusion detection[1], and the event management provided by the CERT, etc. These efforts deal with systems as they are deployed, concentrating on detecting and fixing bugs, rather than on changing or influencing system principles and design.

There seems to be a lack of communication between these schools, and a resulting dearth of pragmatic but structured security solutions. This paper suggests some areas where an integration of these two approaches may provide some insight, and discusses some areas of current success. It does not reject current security models, paradigms, or methods of working, but rather points out some of their limitations and proposes remedies.

1997 New Security Paradigms Workshop Langdale, Cumbria UK

The paper presents new security paradigms on two levels. First, it suggests an integration of two schools of research, a new methodology for solving security problems with an architectural approach, not completely general and not specific to a single implementation. Second, it highlights some areas where new approaches are needed to meet current problems, and it suggests some avenues for analysis and synthesis of solutions.

Limitations of the Formalist Approach

The use of security models and frameworks has been very much of a specialty area, divorced from mainstream system design and operation. This is due to a number of factors, including the way models are formulated and presented. The emphasis is on precision, so that formal reasoning can be applied to show the correctness of the system specification and design with respect to general security properties. Much less effort has been applied to the more difficult to formalize problem of understanding what the desired properties are for a specific system. The formal models emphasize two areas of security: access control enforced by operating system kernels and security protocols. In other areas of security, there is a desire for more assurance, but little support from well-understood and agreed-upon policies, principles and structures.

Secure Systems are not being built

The major effort of the formalists has been in the area of secure operating systems, leading to secure computer systems. The primary security abstraction in this area has been in the area of access control, enforced by a small trusted reference monitor that has been shown to be "correct" with respect to the model. The users, and programs operating on their behalf, are "subjects;" the system controls their access to "objects," which are containers of data. In systems enforcing MAC (mandatory access control), the mechanism enforces a lattice model of user privilege and data classification that is independent of the untrusted parts of the system. This abstraction is attractively simple and general purpose, especially for the type of time-sharing, user driven systems of the 1970s, when the security modeling effort began. Various formalizations of this model[2 ,3] have been created and used in system developments, and it is the basis of most of the security evaluation criteria[4] and secure system designs to date.

However, this effort has had little effect on the security of most systems used today. One of the main premises in this effort is that security must be designed in from the start, not added on; in fact, the access control is generally enforced by a small part of the

operating system, the security kernel. Today, we have well-established commercial operating systems, with a large and growing investment in applications software. These systems were designed for commercial appeal and functionality, not for security. In some cases, they have been augmented or modified to meet security evaluation criteria, but then the commercial version continues to evolve and the evaluation becomes out of date. In addition, for both economic and political reasons, manufacturers are not willing to produce systems secure enough to conform to the formal security definitions, especially Mandatory Access Control (MAC).

The research into secure operating systems has produced significant understanding of secure design, but has not produced secure, commercially viable systems.

A smaller focus with good effect

The newer efforts in protocol analysis are having more impact on current design[5]. It is significant that this work attempts to produce and show correctness of a small, well-defined part of an information system, rather than the system as a whole. The area of authentication and cryptographic protocols is recognized as critical to commercial use of the Internet and other information networks, and the efforts of the formalists are (generally) appreciated and fruitful.

Between these extremes of secure systems and secure special-purpose protocols, there is a gap. Architectural analysis and synthesis are needed to discover and address issues such as appropriate use of authentication protocols in a larger system context.

Limitations of the pragmatic approach

The intrusion detection community, the CERT, and other pragmatic security practitioners perform a critical service for the Internet users and providers by detecting misuses and recommending fixes. However, they deal with the status quo, not with new designs or architectures. Moreover, their analyses focus on specific flaws in specific systems, as they must. There is little effort in this community towards new security architectures or even mechanisms. Their focus is on managing the current crises, of which there are an ample supply. Some of the intrusion detection tools do embody an understanding of more general patterns of vulnerability and abuse. These, as well as the anecdotal data of attacks, can give reality to the analysis suggested above.

Karl Levitt[6]and some of his students have characterized UNIX vulnerabilities and have a program that detects suspicious behavior in UNIX systems. This behavior is "legal," at least in some circumstances, but it has been exploited to attack UNIX systems. An example is root privilege, needed by many system programs to access files and directories, but a popular way to attack UNIX systems.

The UNIX security work has led to the fixing of numerous bugs in the system and to many advisories on system configuration and operation. The basic design of the system has not been affected.

The UNIX intrusion detection work exposes a deeper question: Is it necessary, or even good design, to base the privilege of a program on the privilege of the user "running" it? The type enforcement in the LOCK[7] program, and the Clark and Wilson model[8] address this issue to some extent, but also have not led to new design.

The missing link between these schools

The formalists have had major impact on the design and evaluation of secure systems, though their effect is waning. Because existing systems are far from being either secure or correct, however, their work has not encouraged or even allowed the use of COTS, and especially not the new and constantly changing commercial products. Efforts on composition of systems have emphasized the bad effects of composing good components, rather than the possibility of making trustworthy systems out of untrustworthy but available components. Meanwhile, systems are being built with little or no concern for security, or with security mechanisms and features included in ad hoc and probably ineffective ways.

The pragmatists see the effects of undisciplined growth, poor design and little security. They see the attacks on the systems and their results. However, their knowledge is seldom integrated into new designs or architectures; rather it goes towards fixing bugs in existing systems. The same basic design flaws remain through generations of releases.

We need new security architectures that are both realistic and structured. We need to understand the relationships between security properties of systems and their current or planned functionality. These efforts can be done only if we can integrate the formal and the experiential. Then our abstractions of security will accurately describe our systems, though possibly with less generality and less precision than the current, more universal models.

The Process: Integrating Understanding

We can use high level abstractions of security properties to understand the experiential data, and we can use experience to refine our understanding of security and produce more secure designs. We can use this interactive and iterative process to develop security solutions that are both structured and flexible, and that can address the evolving functions and systems of information technology.

The integrated understanding of the high level abstractions and the detailed observations can sometimes be captured in an architectural approach. This intermediate level of synthesis is useful to describe general but not universal countermeasures that prevent classes of observed attacks from being successful in disrupting systems.

Architectures can capture the physical components of a system, with elements such as servers, clients, firewalls, etc. Architecture can also refer to protocols, specifying layering, peer or client-server interactions, or the state space of a particular protocol. Architecture can also refer to the allocation of functionality within a system, including security functionality, and including the use of security mechanisms that reinforce each other to provide resistance to attacks (security in depth). It is not necessary to think of a system as being totally described by a single

architecture; systems can have all of the kinds of architectures described above. The point is to see some general structure in the system, expressed in terms of its design components or requirements. Within this structure, one can see critical security areas: places where attacks have previously been detected, places where critical functionality is located, places where an attack could damage critical system resources. Rigorous security engineering methods, including formal analysis, can be directed to those aspects of the system where there is the most need and the most potential payoff.

System architectures are seldom determined by security requirements; functionality and the use of existing components generally take precedence. Security analysis of the architecture can help us understand which parts are security-critical, and it is in these areas that we may be able to affect the design. The examples in this paper illustrate security's place in the system. We have had little impact in areas like operating systems, which are central to all system requirements. We have had more impact in smaller, security-critical areas such as cryptographic protocols.

Example Problems

These examples illustrate several different aspects of the security problem, areas where interactions occur or where the models and policies we have do not accurately reflect the systems we build. They also illustrate what is meant by an architectural approach that is general but not universal.

Authentication and access control

Authentication and access control interact. Correct access control decisions cannot be assured when the identity of the requester is not certain [9]. Most of the formal work in access control has dealt with the separation of information within a system, distinguishing among authorized users and preventing data belonging to a user or marked with a sensitivity label from leaking. The systems with strong access control are designed to enable partitioning of the information and control sharing within the system. If there is misauthentication of privileged users, the access control decisions cannot be effective.

The access control models assume completely correct identification of subjects; they focus on the subject-object relationship. Authentication methods yield only a probabilistic identification (though the probability can be very high in some cases), leaving a residual risk. The interaction between the authentication uncertainty and the access control policy is often not even analyzed.

A different situation is posed by the use of strong authentication to allow commerce over the Internet. In this scenario, the user is strongly identified to the system. If these systems do not have strong access control within them, however, there is a risk that users can claim more privilege than they are entitled to, by breaking the internal boundaries. For example, a shopper might get access to other shoppers' card numbers, if they are stored in the system without sufficient protection. Or someone who had paid for a certain level of access might get more, again by breaking internal boundaries. Interactions of this kind need to be addressed so that the security mechanisms for authentication and access control can be effective, not misleading.

Object protection and mobile code

In the usual (TCSEC) model for access control, subjects are active entities that have access to passive objects. This model does not distinguish clearly between code and data, nor does it uniquely distinguish execute access from other types of access. Most of the code in the system is considered untrusted; it is constrained not to leak data, but otherwise its actions are ignored.

The Clark and Wilson model does make a clear distinction between code and data. It, however, assumes a few (large?) programs, which can be called by authorized users to act on appropriate data.

We now have systems (e.g., JAVA[10]) that send and receive large numbers of small software programs via the Internet or other networks. These are run interpretively, but they do cause execution of system functions. The question is how to constrain access to these programs by users, how to import and run these programs safely, and how to protect the programs, users and systems from each other. In this model, it is hard to find a passive entity. The security architecture for JAVA does exist, but it acknowledged that it has flaws, especially in the area of confinement. Can we understand the differences and similarities between the mobile code model and our older security models? Can we develop a structured approach to securing these systems?

It is interesting that the formal models for operating system security assume a subject-object model, and the evaluation criteria essentially require a model of this sort. It is also interesting to notice that the intrusion detection work has found problems with the identification of user and software privilege in cases where the software requires more privilege than the user is normally entitled to. Attacks are already being reported that use mobile code agents and exploit the privileges that are necessary for the programs to work. This may be an area where our experience could lead us to understanding significant security risks and possibly developing some more secure architectures.

Internet Security

About ten years ago, we at GTE developed an architecture for Internet Security[11]. This architecture was based on the DoD (or ISO) protocol layering. The most critical portion of this architecture was in the IP layer of protocol, responsible for the delivery of packets between end systems. We were aware even then of IP spoofing and its dangers. The requirement for security at this layer was called the Internet Security Service (ISS). This included protection, on a per packet basis only, in line with IP. Required protection was confidentiality, packet integrity (ability to detect modification) and packet source authentication. With the ISS, TCP and other higher level protocols could add service like integrity of the data stream and detection of missing or delayed data. The only real service we considered at higher levels was message security, which was special because the messages had to be protected while in storage at an intermediate system. End-to-end encryption was the preferred method for providing both the ISS and message security, since this reduced the security demands on intermediate systems and lowered the risk.

This security architecture was introduced into the Secure Data Network System (SDNS) program[12], and is now reflected, after a long history of change and rediscovery, in the IP/Sec standard. With this kind of protection between systems or between

Intranets, the risk of network attachment can be significantly lowered.

Most data traveling the Internet is not end-to-end encrypted. Most systems do not have hardware encryption devices, and cryptographic key management is still a difficult problem. Cryptography is being used to protect sensitive pieces of information, such as passwords and credit card numbers. In this changed environment, what kind of security architecture makes sense? Some kind of authentication is required at the IP level to thwart IP spoofing attacks, but with what mechanisms and to what degree? We need to update our approach so that it is effective, feasible and suitable for the way the Internet is evolving.

Conclusions

The pragmatists have and are collecting large amounts of information about how systems do and do not work. We need to develop abstractions of security based on this information, giving us an understanding that is formally rigorous and that also accurately describes the security problem.

An integration of the formal and the pragmatic will:

- Structure functional security architectures to protect required system functions;
- Identify critical dependencies and functions, allowing the use of formal methods and new design where they are most important to improve security;
- Facilitate the use of existing system components in trustworthy systems, with the recognition that most systems cannot be custom-made or even redesigned.

These results will provide system designers with incentives to improve security in systems without undue redesign or compromise in system functionality. Only then will we security professionals be able to have significant impact on systems as they change and evolve, since then our solutions will fit the problems.

References

1. S. Kumar, *Classification and Detection of Computer Intrusions*. Ph.D. Thesis, Purdue University, 1995.

2. D.E. Bell and L.J. LaPadula, "Secure Computer Systems: A Mathematical Model," *ESD-TR-73-278 Volume 2*, MITRE, November 1973.

3. J.A Goguen and J. Meseguer. Security Policies and Security Models. Proceedings 1984 IEEE Symposium on Security and Privacy, April 1982

4. *Department of Defense Trusted Computer Security Evaluation Criteria*, DoD 5200.28-STD, National Computer Security Center, December 1985.

5. C. Meadows, "Applying formal methods to the analysis of a key management protocol," *J. Computer Security*, 1(1):5-36, 1992

6. C. Ko, K. Levitt, and G. Fink, Automated detection of vulnerabilities in privileged programs by execution monitoring." In *Proceedings of the Tenth Annual Computer Security Applications Conference*, December 1994.

7. O. S. Saydjari, J. M. Beckman, J.R. Leaman, "Locking Computers Securely." In *Proceedings of the 10th National Computer Security Conference*, Baltimore, MD, 1987.

8. D. Clark and D. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, Oakland, CA, April 1987.

9. R. Nelson, D. Becker, J. Brunell and J. Heimann, "Mutual Suspicion for Network Security," *Proceedings of the 13th National Computer Security Conference*, Baltimore, MD, September 1990.

10. T. Lindholm and F. Yellin. *The Java Virtual Machine Specification*. Addison-Wesley, Menlo Park, CA, 1997.

11. R. Nelson, D. Becker, J. Heimann, J. Sonsini, "Internet Architecture for the DoD" [SECRET], presented at MILCOM 88, October 1988, San Diego, CA.

12. R. Nelson and J. Heimann, "SDNS Architecture and End-to-end Encryption," presented at CRYPTO '89 and published in Lecture Notes in Computer Science 435, Springer-Verlag, 1990