Journal of Computer Security 4 (1996) 233–238
IOS Press

Foreword

Leonard J. LaPadula

1. About the Bell–LaPadula model

Much has changed in computer security since 1973, when the Bell–LaPadula model was first published. Development has moved away from kernel-based secure systems toward application-level security capabilities, providing secure transactions on top of commercial, nonsecure operating systems. The potential for widespread use of encryption for privacy, authentication, and other services, as embodied in digital signature and other technologies, has become quite apparent. At the same time there has been waning interest in formal, or even informal, modeling.

This trend has obvious benefits for electronic commerce, rapid fielding of new systems, reduced cost, and so on, but something important appears not to be happening. The development of information policy from a number of aspects is not getting the attention it deserves. As several colleagues and I have noted at symposia and in papers, there are perhaps seven levels of elaboration of requirements relevant to use of automated systems in enterprises. One view of these identifies

1. Enterprise description
2. Trust objectives
3. External-interface requirements model
4. Internal requirements model
5. Rules of operation
6. Functional design
7. Hardware and software description

Early models of computer security were narrow in their scope, focusing on access control mechanisms within a computer system, at or near the fourth and fifth levels of elaboration. This is certainly the case with the Bell–LaPadula model, the security properties being at the fourth level and its rules of operation being at the fifth level. Since then requirements have been specified at each of the levels above, but I know of no comprehensive set of requirements spanning all levels for a single enterprise or class of enterprises. There is much uncharted territory, apparently few explorers.

The utility of carrying out successive lower levels of elaboration of requirements, starting, say, with the fourth level, is self-evident since it leads to implementations that presumably better match higher level needs than would otherwise be the case.

Some in the profession have argued, starting around the late 1970s, that specifying requirements at the high levels, one through three, is not only desirable, but really quite necessary if what the computer does is to have a sensible and useful relationship to the real world. It may be tempting to underestimate the need for higher level requirements; after all, businesses have been operating successfully for centuries without the aid of enterprise models. However, the computer as employee is a recent phenomenon and computers as yet have neither volition nor innate intelligence. The computer must be told what to do and, to be effective, what it is told must derive from its employer's business.

The several stages of elaboration of requirements provide linkage between the employer's business and useful work by the computer. We used to expect a secure computer merely to enforce access rules for confidentiality; now we should expect a sound computer's behavior to relate in useful ways to its environment. It should be a partner in business,

- providing accurate information from correct inputs;
- creating information of high utility through its presentation methods;
- ensuring availability of information;
- protecting the confidentiality and privacy of information when appropriate;
- safeguarding the timeliness of information; and
- participating in error detection and correction.

It seems to me there is much work to be done here; some, I think, can usefully take the form of modeling.

The Bell–LaPadula model, of course, has its shortcomings, and these have become widely known in the professional community, even to those who have never read the original papers. It is not my purpose to examine nor attempt a defense of those shortcomings. Rather, I will examine one or two characteristics of the model to see how they may relate to issues in automated information processing in 1995.

To set the context of my remarks, I must point out that I will stay within the confines of the model as presented in Volumes I and II of the original papers. In those volumes we basically find, after some introductory material to characterize the modeling, three information-processing principles and ten rules of operation.

The three principles are

- Security Principle – a permitted access by a Subject to an Object satisfies the security principle if either the access does not involve 'reading' the Object (the ability to learn what the Object contains) or the clearance of the Subject is adequate for the classification of the Object.
- Interactivity Principle – the set of all currently permitted accesses by a Subject to a set of Objects satisfies the interactivity principle if the classification of each Object that the Subject can put data into equals or exceeds the classifications of all Objects that the Subject can read.

- Tranquillity Principle – the automated system satisfies the tranquillity principle if the classifications of Objects do not change during normal operation of the system.

The security principle and interactivity principle gave rise to the more commonly known simple security property and *-property, respectively. The names of the ten rules are mostly self-explanatory.

1. get-read
2. get-append
3. get-execute
4. get-write
5. release-read/write/all/execute
6. give-read/write/all/execute (change permissions)
7. rescind-read/write/all/execute (change permissions)
8. change-f (change classification mapping)
9. create-object
10. delete-object

The 'get' rules give a subject the requested access to a referenced object if (1) the subject has permission for the requested access to the referenced object, and (2) the access, if permitted, will satisfy the security and interactivity principles. The 'give' and 'rescind' rules change the permissions of subjects, implementing what is often called a discretionary access control policy. The change-f rule allows the change to the classification mapping if the tranquillity principle is preserved for all active objects. Intuitively, an active object is one that is or might be in use in the system (some subject has one or more access permissions with respect to that object).

The principles are grounded in the problem at hand and, to some extent, take account of the technology as it existed in the early 1970s. The interactivity principle illustrates this. It assumes that a process (a running computer program) in some practical sense 'controls' the information it reads and can do what it will with it. Hence the prohibition against its simultaneously having open a file that it can write or append to and a file of a higher classification that it can read from. The intention clearly is to prevent inadvertent, or even malicious, unauthorized disclosure of the information. But if a process were to have a different character, the interactivity principle might well not be relevant to it.[1]

[1]We might imagine, for example, a process that does not directly manipulate data, but rather makes requests of some 'trusted' agent within the computer. A message-handling process which can 'see' only addresses and classifications and cannot modify them would not need to be governed by the interactivity principle. We might interpret its 'read' as a 'receive' request and its 'write' as a 'transmit' request. The underlying message transfer agent providing the receive/transmit service would presumably either be trusted not to mix data of different classifications or would be subject to the three principles through the intervention of some more trusted control process. But the application-level message-handling process would not be enjoined from simultaneously having receive and transmit access to messages of different classification levels.

The years since the principles were initially articulated have shown them to have been reasonable, useful constraints for protecting confidentiality of classified information. They specify desired conditions that should pertain throughout normal operation of an automated system.

That limited success suggests that perhaps one can state principles for other aspects of information and the enterprises that use it. We have seen such principles in the late 1980s, oriented toward integrity of information and stated at a higher level of abstraction than the Bell–LaPadula model. Ongoing today are efforts to carry those principles through the successive stages of elaboration of requirements, leading, one hopes, to automated systems that can effectively deal with integrity of information in several useful aspects. Ultimately, to make the computer a productive, cooperative partner in an enterprise, the principles must be carried down to some embodiment of policy that the computer can apply in its workings. Here the Bell–LaPadula model provides one paradigm for incorporation of policy – its ten rules of operation.

The rules are an integral part of the Bell–LaPadula model. One can define a model without the rules, but the result is not the Bell–LaPadula model intended by its authors. The objective was to provide a plan for building a secure system, not just to characterize theoretically what information-flow conditions a secure system should satisfy. Thus, a large part of the model results from constructive engineering designed to specify, by rules of operation, how the automated system can preserve the desired conditions expressed in the principles.

The rules give a specification not far removed from the computer program coding that might implement them. This further illustrates the fairly narrow range of specification levels of the Bell–LaPadula model. The principles are framed at the fourth level of elaboration of requirements, and the rules elaborate those principles at the next, fifth, level. That this is so has been realized, at least implicitly, by critics of the Bell–LaPadula model over the years.

The rules embody all three security principles, even though in the mathematical model only preservation of the security and interactivity principles was proved. That they satisfy the tranquillity principle is self-evident. But the rules also constitute an operational policy, about which nothing was proved in the model. Yet, it may be critically important to know what that operational policy is and how it relates to the activities of the enterprise that employs the computer. Is the operational policy useful? Is it relevant to the business of the computer's users?

The theorems of the Bell–LaPadula model tell us that the rules preserve the security and interactivity principles, but they say nothing about the utility of those rules. There is no guarantee that any rule will ever respond with anything more useful than 'No, your request is denied!' So, one can prove that a useless set of rules will give a 'secure'[2] system. Convincing oneself that the rules are useful lies

[2]In this context 'secure' means 'preserves the security and interactivity principles'.

outside the formal model specified in 1973, but it need not lie outside the scope of formal modeling. Given that there are only ten rules in the Bell–LaPadula model, that they were designed in good faith to maximize operational utility, and that the principles to be satisfied were quite simple, the problem of utility could be overcome by carefulness and a bit of luck. In a more complex informational and operational situation, however, one would prefer to have a firmer technological basis for assurance that one's automated system will have high utility in its intended environment.

2. How it happened that we published Volumes I and II

It was June 1994 and the Computer Security Foundations Workshop VII was convened at The Franconia Inn in New Hampshire, USA. During dinner one night, Li Gong suggested to me that the Bell–LaPadula model should be published in an archival journal to provide easy access. Being about as vain as most people, I responded enthusiastically to this patently wonderful idea. Jon Millen was at our dinner table and suggested that the *Journal of Computer Security* might be the appropriate place to publish. We talked about it some, but, inevitably, the table talk turned to other interesting subjects; a large contingent of attendees had been to see the newly released movie Jurassic Park.

Upon sober reflection on this marvelous idea, several questions occurred to me, not the least of which was "Who needs access anymore?" Another was "Which model are we talking about?" And, finally, "Why not just put it out there on the Internet?" Eventually, after several conversations over several months, I decided where I stood on the questions. First, I firmly held the opinion that the best representation of the Bell–LaPadula model is in Volumes I and II of the original technical reports. Subsequent publications in the series provide refinements and particularize the model to a specific computing environment. My belief is that understanding of the fundamental modeling approach and the model's structure best comes from Volumes I and II. On the question of "Who needs access anymore?", I am still not sure, but I think some people may have an historical or research perspective that might be served by having easy access to the original representation of the model. Finally, it seemed to me that producing an electronic copy for publication in an archival journal would be a good step toward making it available on the Internet.

Thus, Jon Millen and I agreed on a plan for publishing Volumes I and II and this Foreword in this issue of the Journal. Having settled that, I started exploring ways to create an electronic version. I tried scanning the original copies and fixing up the results a bit, thinking that approach would be quick and easy. Not so. The technology for scanning and optical character recognition is good, but not good enough to deal with all the special characters and formatting you find in the original papers. It turned out to be easier to recreate the originals using a capable word

processor. It was time-consuming, but it had its good side: I became an expert on the Bell–LaPadula model! Along the way, I found and corrected (I hope) about seven typos.

Chelmsford, September 1995