Essay 5

Abstraction and Refinement of Layered Security Policy

Marshall Abrams and David Bailey

There are multiple views of corporate (enterprise) computing, each with its own metaphors and terms of reference. The different views incorporate different levels of abstraction, in which details are suppressed to concentrate attention on the issues important to the particular observer. This essay examines these different metaphors with respect to the enterprise security policy resulting in a layered policy where each main layer relates to one of the system metaphors and the policy described for a lower level of detail is an implementation of the policy at a higher level. The layered view of policy helps system designers, managers, and users understand the rationale for security policy at the lowest levels of abstraction, because the relationship of the low level policy to the enterprise information policy is clear.

Levels of abstraction and policy

There are multiple views of corporate (enterprisecomputing, each with its own metaphors and terms of reference. The metaphors with which we view computing differ from level to level of abstraction. Abstraction serves the very useful purpose of suppressing details not of interest to the observer. Suppressing these details makes it possible to concentrate on the issues that the observer considers important. One observer may think of corporate information processing resources, another of the "operating system," and another of the "network." The terms of reference are different, the concerns are different, and the policy statements may appear different. But the policy and rules must be consistent at all levels of abstraction for the organization to achieve the protection desired.

This essay has two primary purposes. The first is to exhibit several different ways in which enterprise computing is viewed by different members of the enterprise. We observe that all of the enterprise members are

talking about the same computing structure when they talk about computing, even though their interests and their language might suggest otherwise. Furthermore, each member of the enterprise has a view of the enterprise's policy for protecting information, and, once again, they are all talking about the same thing, even though their language might suggest otherwise.

The second purpose is to observe that when we choose the members of the enterprise appropriately, their views of computing can be seen as successive refinements of the high-level view taken by the CEO. Because the refinements cover a very broad range and use more than one metaphor, the nature of the layers as refinements is a little harder to see than it is in a decomposition that covers a narrow range. The structure of the policy is, again, parallel to the computing structure; the views of policy held by members of the enterprise constitute successive refinements of the policy expressed by the CEO.

Each of the many views of computing within an enterprise reflects its holder's responsibilities and relationship to the computing resource. What is important to workers at one level may be incomprehensible detail to those at another level. Conversely, a statement that is clear and complete at one level may appear vague and general to those at another level. While many views are possible, some are better suited than others to our primary purpose. The views or levels we will discuss include:

- Top management's view of computing as an information and resource management problem.
- The computer user's view of computing functions and data as resources available to support some task.
- The system builder's view of the computing system as physical components in specific locations delivering various services to users.

Both the complexity of modern computing and the different interests of various people in the enterprise warrant the use of different metaphors to provide a complete discussion of the computing system. This complexity will be reflected in the enterprise security policy which must be discussed in terms of all metaphors. The result will be a layered policy where each main layer relates to one of the system metaphors. All views must fit together in the sense that the policy described for a lower level of detail must be shown to be an implementation of the policy at a higher level.

As one approaches a computing system from afar (that is, with progressively less abstract logical views), one sees a succession of different aspects of the same object. From great distance, one might see computing as an information management function of the enterprise. Moving closer, one might see a collection of services used by people in the enterprise and running on some sort of "computer." Still closer, the view might be of

a network of computers providing different services in different physical locations, on different hardware, to different sets of users, connected in different ways. Even though the metaphors used to discuss the various layers differ, each can be seen as a refinement of the previous layer, including more detail and using a different language, but still discussing the same "computing system."

If the enterprise has any interest in protecting its information assets against undesirable events (and it is difficult to imagine one that doesn't), then it has (or should have) a policy for protecting those information assets, otherwise known as its information security policy. The policy should address the information assets of the organization, the threats to those assets, and the measures that management has decided are reasonable and proper to protect those assets. Management may base its decision on cost-benefit analysis by weighing the cost of expected (or experienced) losses against the cost of preventative measures. When cost analysis is not possible or believable, a policy basis may exist for protective measures. And in some cases management will have to make decisions based on insufficient information. Top management will usually be assisted by many other employees in reaching a decision. Policy statements will be prepared by subordinates for approval by top management. It is this approval that puts the policy into force.

We have one enterprise-wide computing system viewed from different perspectives with the aid of different metaphors or abstractions. The top management sees information and corporate functions. Some users and management see files and services on a computer; others see a network. For many organizations, computing is large and complicated, and we accept that different people with different interests have different views of it. We should not expect that a single view of security policy will suffice for all people. For example, the top management will probably not understand the terms of reference at the network level of abstraction and will not care about movement of bits from one network machine to another, as long as the information management policy is satisfied.

However, even though different people use different terminology, there is a need for a single unified policy (otherwise, someone has not been listening, and the top management has lost control). It is probably created by an iterative process of composition and decomposition. In the process, some of the policies at more detailed levels will have to be changed to conform with the highest level enterprise policy as it develops. Over the years the policies at various levels will have to be changed to accommodate new modes of operation, new insights, and new organizational concerns. Probably the various levels of policy will never be 100 percent synchronized at any instant of time.

Policies at different levels of abstraction must be connected. Policy is composable and decomposable in the same way as system design. At various levels of abstraction, the metaphors and terms of reference on

which the policy is based change, but the levels remain connected. It must remain possible, even across changes in metaphor, to demonstrate that each lower level implements the level above it.

Three views of computing and security policy

Top management view. At the top level of abstraction, the interests of the CEO and the board of directors are in management and protection of corporate information. Computing may not even be an interest except to the extent that the computers themselves represent an asset and a cost, and the ability to compute represents a corporate resource. In a modern information-oriented corporation, the CEO may even have a view of the things that are done with information within the enterprise and the protection that must be afforded to various types of information. These views will be described in information terms, not in computing terms.

The policy for information management in the enterprise will similarly be expressed by (and for) the top management and the members of the board. It will cover the protection needs of the computing function viewed as a function within the enterprise, expressed as rules identifying information security objectives and delineating approved and unapproved behavior. At this high level of abstraction, all policies merge into something like "Protect the organization's information assets." While it is easy to understand the sentiment behind such a generalization, it does not provide much guidance on whether specific actions are permitted or not. Introducing such specificity at lower levels of abstraction requires interpretation of the high-level policy. Specifically, it requires an approval process (or delegation of approval authorit) so that the implementation is scrutinized and approved by the upper level management.

Individual users' views. Moving to a lower level of detail, individual computer users using specific services become visible. The users have two different views of computing, depending somewhat on individual views of what they are doing. Some users think in terms of specific functions that they perform using a computer: "I process purchase requests using the computer." Others think in more machine-oriented terms: "I use the Dragon machine when I process purchase requests." We coalesce these two views into a single view in which users use "large" functions of the system, such as application programs.

At this level of detail, we see controls on the use and modification of data and on the way the services themselves are modified. These rules are imposed to satisfy the requirements created by the top management's policy. They may control who can view proprietary information. They may control who can modify high-integrity information and the circumstances

under which it can be modified. They may be imposed in an attempt to maintain availability of data or services.

This collection of rules together constitutes a user access policy regulating the access of various users to various objects. This is only part of an information security policy; additional asset-protection information is necessary in such policy elements as physical and administrative security, backup procedures, disaster recovery, and so on.

The user access policy is an implementation of the top management's policy. The people concerned with the specification, design, and implementation of the user access policy have a different view of computing than the top management. Their interests and concerns are different based on a greater level of detailed knowledge of computing. These facts, however, do not permit them to create a new policy — they are expected to implement the policy adopted by management. And they will be expected to make a convincing argument that the policy regulating human user access to computer programs and files is an implementation of management's information policy. It must be an implementation in the sense that it does not permit computer users to take actions that violate the information policy.

Many organizations provide guidance on how to translate the top-level information security policy into a user access policy. This guidance is often called "implementing guidance," "implementing instructions," or something similar. Organizations with many hierarchical levels may provide implementing guidance at each level.

Process-level view. The next level of detail down from the user level can be described as the process level or the network level. At this level of detail we find the mechanisms that are invoked to satisfy user requests. Data units are messages or perhaps buffers full of bits rather than files residing on some disk. Here we see for the first time processing that is not directly associated with some human user. These processes are part of the management of the computing resource itself or are present to satisfy user requests for data.

The differences between the process level and the user level occur in several areas. The time scale is much shorter on the process level. At the user level, we talk in terms of requesting and obtaining services, items that are measured in seconds and larger units. At the process level, we talk about requests satisfied on scales of milliseconds or less. The units in which processes use data are much smaller at the process level. Processes deal with requests, messages, and buffers; human users deal with files or application programs.

When considering protection, however, the biggest difference is in the static character of the user level versus the dynamic nature of the process level. At the user level, access to information can be characterized using a finite-state machine in which a system state consists of a de-

scription of the access rights of every human user to every controlled object. A state transition is an atomic action changing the access relationship between one user and one object. At no time does a human subject "kind of" have access to any object.

At the process level, however, this clean state machine or "access matrix" model, as it is sometimes called, doesn't work very well. When a user is granted access to a file on a disk, the bits in the file have to be moved from the storage location into a buffer belonging to the user's surrogate process, where they can be used. At any given time after access is granted, some of the bits will be in the process buffer, some on the disk, and others in buffers in other locations. Depending on the complexity of the system, there could be many buffers in many locations. The undelivered bits will be delivered if requested, but this is not justification for asserting that the user has access to them any more than it was reasonable to claim the user had access to the file before the access was requested. At this level of detail, file access is not atomic — the user has partial access to the file.

Housekeeping and internal service processes introduce a complexity in the user access policy because access rights cannot be traced to an individual user. This dilemma is often resolved by identifying such processes as belonging to the system operator or security authority, as appropriate. This generally produces confusing and unsatisfactory results because thinking about what the human occupants of these roles should or should not do leads to inappropriate controls on the system. In the office analog of this situation, we carefully regulate who can use which information and what they can do with it, but we forget that the secretaries see everything. Declaring that the secretary is a security officer will not, by itself, improve our confidence that data is appropriately protected.

Technical policy models

Policy can be expressed in three different forms: natural language statements, mathematical or nonmathematical formal statements based on a model, and computer implementation mechanisms. Since natural language is prone to ambiguities and multiple interpretations, some means is necessary to determine whether a specific behavior constitutes a violation or attempted violation. In information security, there is an attempt to reduce the ambiguity by basing the policy on a policy model. If the policy model and the system-specific statement of policy are expressed in a mathematical form, the ambiguity will be reduced substantially. For more information about models, see Essay 8.

Expressing the policy models in a commonly understood notation may be helpful in many ways. Using this common notation to express the security policy models for each concern may make it easier for those people with mathematical intuition and experience to identify commonality

among policies as well as issues within each policy. They may also compare alternative models to find the model(s) they consider best (according to some criteria not discussed here). Other people may find it easier to identify commonality in the mechanisms. Both sets of skills can contribute to the security of a real system.

Formal expression of policy

As we have said, formal expression of policy is used to reduce ambiguity. The expression can be either mathematical or nonmathematical. If it is done mathematically, the ambiguity inherent in natural language can be reduced to a minimum. Other advantages include a variety of tools that can be applied to find mistakes in the policy expression and to prove that the policy, if correctly implemented, will have certain desirable properties. The cost of specifying policy in this way is that far fewer people will be able to read the results. Generally, everything having to do with the policy will have to be done by specialists. In a practical sense, this may reduce the likelihood that the policy is correctly implemented, since the specialists will almost certainly not be the people writing the code.

A nonmathematical, but constrained and more precise form of natural language will inevitably be more ambiguous than a mathematical version. Such a policy expression is also less amenable to manipulation by supporting tools, so it will be harder to show that it is self-consistent and has the properties desired. On the other hand, more people can participate in its construction, and more people can read and understand the results. This means there is a higher likelihood that it expresses the desired policy and that it will be correctly used to implement the system. Which version to use depends on the degree of assurance of correctness that is required.

Each of the three policy levels we have discussed can be formalized. Formal models have been produced many times at the user access control level. There is less experience with formal models at the process or network level, but such models have been built as well. It is also possible to formalize a model for policy at the information handling level, although this is rarely done. For example, corporations keep various information about their employees. Consider just two categories of information — salary and health insurance — and the policy governing read access to the information by various people. An employee has access to all of his own information, but no information about other employees. A supervisor has access to salary information about all the people she supervises, but to health insurance information for none of them. A benefits counselor has access to all health insurance information, but no salary information. Notice also that all of the people with special roles are also employees — their status is governed by some combination of rules (see the section below on multiple policies).

The entire collection of rules in the example above could be written down in a precise form of English or could be specified using any specification language at least as rich as set theory. Either form would constitute a formal expression of a specific policy. The policy and many others that could be obtained by varying some of the rules are derived from a model based on people with specific roles accessing identified kinds of information.

Each of the policy levels seems to have a natural model. Not surprisingly, they are different. At the top level, a model can be built using people who are members of groups (probably based on the organizational roles the people play). Groups have information that they share internally, and they have some policy for sharing with other groups.

At the next level of detail, computer users with privileges access files with attributes. The policy is about (semi-) static relationships between users and collections of data. To maintain data integrity, the policy might make statements about which privileges are required to write or modify specific collections of data. The metaphor differs from that of people sharing information within groups, but it is not too difficult to see how to construct a mapping from one metaphor to the other. Such a mapping will be needed to demonstrate that the access controls provided for data files implement the policy of sharing information within groups.

At the process or network level of abstraction, packages of bits are moved about based on attributes of processes or storage containers for the bits. We can see two different kinds of process: those owned by users (that is, they didn't exist before the user took some action to directly or indirectly create them), and those that are part of the system (they predate any user sign-on and continue to exist after all users have signed off). The bits moved are generally only a part of something that has meaning to a user, and they have no meaning to the processes moving them. The natural expression for policy at this level of abstraction is to control the flow of bits between processes based on the type of the process (system or user), attributes of the bits (taken from a source process or container), and attributes of the medium over which the bits are to flow.

As is the case between the computer user level and the information management level, it is necessary to map from the abstraction of the lower level to those of the higher levels. This mapping is necessary to support the argument that the lower level is an implementation of the higher level.

Mapping between levels of policy. In general, the technical policy rules will not be an exact implementation of the high-level prose abstraction of the information policy. It may be impossible, or at least exceedingly difficult, to implement the natural language policy in rules within the computer. The conservative position is to make the rules more re-

strictive than the high-level policy statement. If the rules are too restrictive in the sense that they make it too difficult, or even impossible, to perform an operation that is necessary for the functioning of the enterprise, then special exemption rules are crafted to permit the desired operation(s).

The test of the connection between different levels of policy is that

1. every access allowed by the higher level policy should be supported by the lower level policy, and
2. no action allowed by the lower level policy should be forbidden by the higher level policy.

The conservative position is

1. that there may be an action allowed by the higher level policy that is forbidden by the lower level policy, but
2. no action or combination of actions allowed by the lower level policy can violate the higher level policy.

It is not necessarily easy to demonstrate the correspondence between the high-level and the low-level policies. The terms of reference are different, and the metaphors on which the policies are based are different. Controls will be implemented to maintain privacy or secrecy or to ensure integrity — whatever the top management told us we are supposed to do. These controls constitute a security policy that is expected to be an implementation of the top-level policy. We show this by showing that it implements the next higher level of policy.

Beginning at the network level, for example, we use the mapping between metaphors to map the policy controls at the lower level into controls at the user access level. If the policy has been constructed carefully, it will be possible to argue that it is exactly an implementation of the access policy in the sense given above. Otherwise, we need to show that it is more conservative than the higher level policy. To be less conservative means that some flow can be found that violates the access control rules, and this is not acceptable. In the same way, we show that the access control rules implement the corporate information management policy. By implication, the flows in the network also satisfy the information management policy.

Multiple policies. A single system may be required to satisfy several different policies covering different subject areas. For example, the system may be required to protect the confidentiality of its data. It may also attempt to protect correctness (usually called integrity) by, for example, limiting who can add to or modify some collections of data. In some situations, system behaviors may be limited because of safety concerns.

Other possible security concerns include completeness, accuracy, timeliness, and availability. Some authorities would include integrity; others would remove availability; most would not include safety. (See D.F. Sterne's paper [STER91] for one viewpoint.)

Every threat that management has decided to protect against must be addressed by a corresponding policy. Trying to produce a model that covers all of these subject areas could result in a policy that is very confusing. A mathematical expression of such a policy would be extremely complicated, and it would be very hard to claim that it actually captures the desired policy correctly. To avoid this, policy modelers typically make a simplifying assumption. They begin by assuming that the security concerns (for example, confidentiality, completeness, accuracy, timeliness, and availability) are independent. This is, of course, a simplification. There may be trade-offs or even conflicts. But it is convenient to assume independence as a first-order approximation. With this simplification, the policies can be modeled separately. The modeling can consider each policy as if it were the only policy in effect.

The effect of all the policies taken together is described by a metapolicy (a policy about policies) that describes how the applicable policies combine. Two simple metapolicies for combining two constituent policies will illustrate the concept. A metapolicy that requires approval of both constituent policies could be expressed "Access is permitted only if the user has the necessary clearance AND the need to access the resource in order to perform his/her job." A metapolicy that requires approval of either of the two constituent policies could be expressed "Access to salary information is granted to a supervisor OR to a member of the compensation administration."

Physical implementation

The implementation of a policy, defined as precisely as possible in a model, uses mechanisms within the computer system. Some mechanisms can be effective in implementing more than one policy. Identification of such mechanisms may be based on intuition, experience, or systematic analysis. There may be a cost savings in choosing to use such a mechanism in an application environment that includes these multiple policies. Selection of mechanisms involves many engineering analysis and design activities. Trade-off analysis among cost, performance, ease of use, storage requirements, and so on, is involved.

Conclusions

We have given examples to show that the single computing entity built by an organization to satisfy its information processing needs is viewed by different members of the organization in different ways. Some see in-

formation management, some see a computer used by users, and some see a network that moves bits around on behalf of users. Each of these collections of people uses different metaphors and different terminology to describe the same entity. We tolerate this diversity because the different collections of people have different needs and interests. Forcing them to speak the same language would interfere with their ability to perform their jobs and serve no useful purpose.

In the same way, there is one policy governing how the computing entity is used to process corporate information. The different collections of people have different metaphors and terminology for discussing the single policy for the same reasons that they have different metaphors and language for discussing the computer system. Again, for the same reasons, we must tolerate this diversity in language. There is one system. There is one policy governing it, and there are many ways to view both the system and the policy.