ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 1, November 2012

Policies and Models of Security

Roopha Shree Kollolu Srinivasa[1]

B. E Student, Department of Computer Science Engineering, East West Institute of Technology, India[1]

ABSTRACT: The categories of separation are listed roughly in increasing order of complexity to implement, and, for the first three, in decreasing order of the security provided. However, the first two approaches are very stringent and can lead to poor resource utilization. Therefore, we would like to shift the burden of protection to the operating system to allow concurrent execution of processes having different securityneeds.

KEYWORDS*:* Network Security, policies, models

I. INTRODUCTION

Protected objects

The rise of multiprogramming meant that several aspects of a computing system required protection:

- ✧ memory
- ✧ sharable I/O devices, such as disks
- ✧ serially reusable I/O devices, such as printers and tapedrives
- ✧ sharable programs andsubprocedures
- ✧ sharabledata

As it assumed responsibility for controlled sharing, the operating system had to protect theseobjects.

Security in operating system

The basis of protection is separation: keeping one user's objects separate from other users.Rushby and Randell noted that separation in an operating system can occur in several ways:

- *physical separation*, in which different processes use different physical objects, such as separate printers for output requiring different levels ofsecurity
- *temporal separation*, in which processes having different security requirements are executed at differenttimes
- *logical separation*, in which users operate under the illusion that no other processes exist, as when an operating system constrains a program's accesses so thatthe
program cannot access objects outside its permitted domain
- *cryptographic separation*, in which processes conceal their data and computations in such a way that they are unintelligible to outsideprocesses

Of course, combinations of two or more of these forms of separation are also possible.

But separation is only half the answer. We want to separate users and their objects, but we also want to be able to provide sharing for some of those objects. For example, two users with different security levels may want to invoke the same search algorithm or function call.

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 1, November 2012

We would like the users to be able to share the algorithms and functions without compromisingtheir individual security needs. An operating system can support separation and sharing in several ways, offering protection at any of several levels.

- *Do not protect.* Operating systems with no protection are appropriate when sensitive procedures are being run at separatetimes.
- *Isolate*. When an operating system provides isolation, different processes running concurrently are unaware of the presence of each other. Each process has its own address space, files, and other objects. The operating system must confine each process somehow so that the objects of the other processes are completelyconcealed.
- *Share all or share nothing.* With this form of protection, the owner of an objectdeclares it to be public or private. A public object is available to all users, whereasa
private object is available only to its owner.
- *Share via access limitation.* With protection by access limitation, the operating system checks the allowability of each user's potential access to an object. That is, access control is implemented for a specific user and a specific object. Lists of acceptable actions guide the operating system in determining whether a particular user should have access to a particular object. In some sense, the operating system acts as a guard between users and objects, ensuring that only authorized accessesoccur.
- *Share by capabilities.* An extension of limited access sharing, this form of protection allows dynamic creation of sharing rights for objects. The degree of sharing can depend on the owner or the subject, on the context of the computation, or on the objectitself.
- *Limit use of an object.* This form of protection limits not just the access to an object but the use made of that object after it has been accessed. For example, a user may be allowed to view a sensitive document, but not to print a copy of it. More powerfully, a user may be allowed access to data in a database to derive statistical summaries (such as average salary at a particular grade level), but not to determine specific data values (salaries ofindividuals).

II. METHODS OF MEMORY PROTECTION

Memory protection is a way to control memory access rights on a computer, and is a part of most modern operating systems. The main purpose of memory protection is to prevent a process from accessing memory that has not been allocated to it. This prevents a bug within a process from affecting other processes, or the operating system itself, and instead results in a segmentation fault or storage violation exception being sent to the offending process, generally causing abnormal termination (killing the process). Memory protection for computer security includes additional techniques such as address space layoutrandomization and executable spaceprotection.

Segmentation

Segmentation refers to dividing a computer's memory into segments. A reference to a memory location includes a value that identifies a segment and an offset within that segment.The x86 architecture has multiple segmentation features, which are helpful for using protected memory on this architecture. On the x86 processor architecture, the GlobalDescriptor Table and Local Descriptor Tables can be used to reference segments in the computer's memory. Pointers to memory segments on x86 processors can also be stored in the processor's segment registers. Initially x86 processors had 4 segment registers, CS (code segment), SS (stack segment), DS (data segment) and ES (extra segment); later another two segment registers were added – FS andGS.

Paged virtual memory

In paging the memory address space is divided into equal-sized blocks called pages. Using virtual memory hardware, each page can reside in any location of the computer's physical memory, or be flagged as being protected. Virtual memory makes it possible to have a linear virtual memory address space and to use it to access blocks fragmented over physicalmemory address space.Most computer architectures which support paging also use pages as the basis for memoryprotection.

A *page table* maps virtual memory to physical memory. The page table is usually invisible to the process. Page tables make it easier to allocate additional memory, as each new page can be allocated from anywhere in physicalmemory.

It is impossible for an application to access a page that has not been explicitly allocated to it, because every memory address either points to a page allocated to that application, or generates an interrupt called a *page fault*. Unallocated pages, and pages allocated to any other application, do not have any addresses from the application point of view.

A page fault may not necessarily indicate an error. Page faults are not only used for memory protection. The operating system may manage the page table in such a way that a reference to a page that has been previously swapped out to disk causes a page fault. The operating system intercepts the page fault and, loads the required memory page, and the application continues as if no fault had occurred. This scheme, known as virtual memory, allows in- memory data not currently in use to be moved to disk storage and back in a way which is transparent to applications, to increase overall

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 1, November 2012

memorycapacity.
software fault handler can, if desired, check the missing key against a larger list of keys maintained by software; thus, the protection key registers inside the processor may be treated as a software-managed cache of a larger list of keys associated with a process.

Simulated segmentation
Simulation is use of a monitoring program to interpret the machine code instructions of some computer architectures. Such an Instruction Set Simulator can provide memory protection by using a segmentation-like scheme and validating the target address and length of each instruction in real time before actually executing them. The simulator must calculate the target address and length and compare this against a list of valid address ranges that it holds concerning the thread's environment, such as any dynamic memoryblocks acquired since the thread's inception, plus any valid shared static memory slots. The meaning of "valid" may change throughout the thread's life depending upon context. It may sometimes be allowed to alter a static block of storage, and sometimes not, depending upon the current mode of execution, which may or may not depend on a storage key or supervisorstate.
It is generally not advisable to use this method of memory protection where adequate facilities exist on a CPU, as this takes valuable processing power from the computer. However, it is generally used for debugging and testing purposes to provide an extra fine level of granularity to otherwise generic storage violations and can indicate precisely which instruction is attempting to overwrite the particular section of storage which may have the same storage key as unprotected storage.

III. FILE PROTECTIONMECHANISM

Until now, we have examined approaches to protecting a general object, no matter the object's nature or type. But some protection schemes are particular to the type. To see how they work, we focus in this section on file protection. The examples we present are only representative; they do not cover all possible means of file protection on themarket.
Basic Forms of Protection
We noted earlier that all multiuser operating systems must provide some minimal protection to keep one user from maliciously or inadvertently accessing or modifying the files of another. As the number of users has grown, so also has the complexity of these protection schemes.

All "None Protection
In the original IBM OS operating systems, files were by default public. Any user could read, modify, or delete a file belonging to any other user. Instead of software- or hardware-based protection, the principal protection involved trust combined with ignorance. System designers supposed that users could be trusted not to read or modify others' files, because the users would expect the same respect from others. Ignorance helped this situation, because a user could access a file only by name ; presumably users knew the names only of those files to which they had legitimate access.
However, it was acknowledged that certain system files were sensitive and that the system administrator could protect them with a password. A normal user could exercise this feature, but passwords were viewed as most valuable for protecting operating system files. Two philosophies guided password use. Sometimes, passwords were used to control all accesses (read, write, or delete), giving the system administrator complete control over all files. But at other times passwords would control only write and delete accesses , because only these two actions affected other users. In either case, the password mechanism required a system operator's intervention each time access to the filebegan .
However, this all-or-none protection is unacceptable for several reasons.

- Lack of trust . The assumption of trustworthy users is not necessarily justified. For systems with few users who all know each other, mutual respect might suffice; but in large systems where not every user knows every other user, there is no basis fortrust.
- All or nothing . Even if a user identifies a set of trustworthy users, there is no convenient way to allow access only tothem.
- Rise of timesharing . This protection scheme is more appropriate for a batch environment, in which users have little chance to interact with other users and in which users do their thinking and exploring when not interacting with the system. However, on timesharing systems, users interact with other users. Because users choose when to execute programs, they are more likely in a timesharing environment to arrange computing tasks to be able to pass results from one program or one user toanother.
- Complexity . Because (human) operator intervention is required for this file protection, operating system performance is degraded. For this reason, this type of file protection is discouraged by computing centers for all but the most sensitive datasets.
- File listings . For accounting purposes and to help users remember for what files they are responsible, various system utilities can produce a list of all files. Thus, users arenotnecessarily ignorant of what files reside on the system. Interactive users may try to browse through anyunprotected files.

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 1, November 2012

Group Protection

Because the all-or-nothing approach has so many drawbacks, researchers sought an improved way to protect files. They focused on identifying groups of users who had some common relationship. In a typical implementation, the world is divided into three classes: the user, a trusted working group associated with the user, and the rest of the users. For simplicity we can call these classes user, group, and world . This form of protection is used on some network systems and the Unixsystem.

All authorized users are separated into groups. A group may consist of several members working on a common project, a department, a class, or a single user. The basis for group membership is need to share . The group members have some common interest and therefore are assumed to have files to share with the other group members. In this approach, no user belongs to more than one group. (Otherwise, a member belonging to groups A and B could pass along an A file to another B groupmember.)

When creating a file, a user defines access rights to the file for the user, for other members of the same group, and for all other users in general. Typically, the choices for access rights are a limited set, such as {read, write, execute, delete}. For a particular file, a user might declare read-only access to the general world, read and write access to the group, and all rights to the user. This approach would be suitable for a paper being developed by a group, whereby the different members of the group might modify sections being written within the group. The paper itself should be available for people outside the group to review but notchange.

A key advantage of the group protection approach is its ease of implementation. A user is recognized by two identifiers (usually numbers): a user ID and a group ID. These identifiers are stored in the file directory entry for each file and are obtained by the operating system when a user logs in. Therefore, the operating system can easily check whether a proposed access to a file is requested from someone whose group ID matches the group ID for the file to be accessed.

Although this protection scheme overcomes some of the shortcomings of the all-or-nothing scheme, it introduces some new difficulties of its own.

- Group affiliation . A single user cannot belong to two groups. Suppose Tom belongs to one group with Ann and to a second group with Bill. If Tom indicates that a file is to be readable by the group, to which group(s) does this permission refer? Suppose a file of Ann's is readable by the group; does Bill have access to it? These ambiguities are most simply resolved by declaring that every user belongs to exactly one group. (This restriction does not mean that all users belong to the samegroup.)
- Multiple personalities . To overcome the one-person one-group restriction, certain people might obtain multiple accounts, permitting them, in effect, to be multiple users. This hole in the protection approach leads to new problems, because a single person can be only one user at a time. To see how problems arise, suppose Tom obtains two accounts, thereby becoming Tom1 in a group with Ann and Tom2 in a group with Bill. Tom1 is not in the same group as Tom2, so any files, programs, or aids developed under the Tom1 account can be available to Tom2 only if they are available to the entire world. Multiple personalities lead to a proliferation of accounts, redundant files, limited protection for files of general interest, and inconvenience tousers.
- All groups . To avoid multiple personalities, the system administrator may decide that Tom shouldhaveaccesstoallhisfilesanytimeheisactive.Thissolutionputstheresponsibilityon Tom to control with whom he shares what things. For example, he may be in Group1 with Ann and Group2 with Bill. He creates a Group1 file to share with Ann. But if he is active in Group2 the nexttime he is logged in, he still sees the Group1 file and may not realize that it is not accessible to Bill, too.
- Limited sharing . Files can be shared only within groups or with the world. Users want to be able to identify sharing partners for a file on a per-file basis, for example, sharing one file with ten people and another file with twentyothers.

Single Permissions

In spite of their drawbacks, the file protection schemes we have described are relatively simple and straightforward. The simplicity of implementing them suggests other easy-to- manage methods that provide finer degrees of security while associating permission with a single file.

Password or Other Token

We can apply a simplified form of password protection to file protection by allowing a user to assign a password to a file. User accesses are limited to those who can supply the correct password at the time the file is opened. The password can be required for any access or only for modifications (writeaccess).

Password access creates for a user the effect of having a different "group" for every file. However, file passwords suffer from difficulties similar to those of authentication passwords:

- Loss . Depending on how the passwords are implemented, it is possible that no one will be able to replace a lost or forgotten password. The operators or system administrators can certainlyintervene and unprotect or assign a particular password, but often they cannot determine what password a user has assigned; if the user loses the password,

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 1, November 2012

a new one must beassigned.

- Use . Supplying a password for each access to a file can be inconvenient and timeconsuming.
- Disclosure . If a password is disclosed to an unauthorized individual, the file becomes immediately accessible. If the user then changes the password to reprotect the file, all the other legitimate users must be informed of the new password because their old password will fail.
- Revocation . To revoke one user's access right to a file, someone must change the password, thereby causing the same problems asdisclosure.

IV. TEMPORARY ACQUIRED PERMISSION

The Unix operating system provides an interesting permission scheme based on a three- level user "group "world hierarchy. The Unix designers added a permission called set userid (suid) . If this protection is set for a file to be executed, the protection level is that of the file's owner , not theexecutor . To see how it works, suppose Tom owns a file and allows Ann to execute it with suid . When Ann executes the file, she has the protection rights of Tom, not ofherself.

This peculiar-sounding permission has a useful application. It permits a user to establish data files to which access is allowed only through specified procedures.

For example, suppose you want to establish a computerized dating service that manipulates a database of people available on particular nights. Sue might be interested in a date for Saturday, but she might have already refused a request from Jeff, saying she had other plans. Sue instructs the service not to reveal to Jeff that she is available. To use the service, Sue, Jeff, and others must be able to read and write (at least indirectly) the file to determine who is availableortopostheiravailability.ButifJeffcanreadthefiledirectly, he would find that Sue has lied. Therefore, your dating service must force Sue and Jeff (and all others) to access this file only through an access program that would screen the data Jeff obtains. But if the file access is limited to read and write by you as its owner, Sue and Jeff will never be able to enter data intoit.

The solution is the Unix SUID protection. You create the database file, giving only you access permission. You also write the program that is to access the database, and save it with the SUID protection. Then, when Jeff executes your program, he temporarily acquires your access permission, but only during execution of the program. Jeff never has direct access to the file because your program will do the actual file access. When Jeff exits from your program, he regains his own access rights and loses yours. Thus, your program can access the file, but the program must display to Jeff only the data Jeff is allowed tosee.
This mechanism is convenient for system functions that general users should be able to perform only in a prescribed way. For example, only the system should be able to modify the file of users' passwords, but individual users should be able to change their own passwords any time they wish. With the SUID feature, a password change program can be owned by the system, which will therefore have full access to the system password table. The program to change passwords also has SUID protection, so that when a normal user executes it, the program can modify the password file in acarefully constrained way on behalf of the user.

V. USER AUTHENTICATION

An operating system bases much of its protection on knowing who a user of the system is. In real-life situations, people commonly ask for identification from people they do not know: A bank employee may ask for a driver's license before cashing a check, library employees may require some identification before charging out books, and immigration officials ask for passports as proof of identity. In-person identification is usually easier than remote identification. For instance, some universities do not report grades over the telephone because the office workers do not necessarily know the students calling. However, a professor who recognizes the voice of a certain student can release that student's grades. Over time, organizations and systems have developed means of authentication, using documents, voice recognition, fingerprint and retina matching, and other trusted means of identification.
In computing, the choices are more limited and the possibilities less secure. Anyone can attempt to log in to a computing system. Unlike the professor who recognizes a student's voice, the computer cannot recognize electrical signals from one person as being any different from those of anyone else. Thus, most computing authentication systems must be based on some knowledge shared only by the computing system and the user. Authentication mechanisms use any of three qualities to confirm a user'sidentity.

1. Something the user *know*sa Passwords, PIN numbers, passphrases, a secret handshake, and mother's maiden name are examples of what a user mayknow.
2. Something the user *has*a Identity badges, physical keys, a driver's license, or a uniform are common examples

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 1, November 2012

of things people have that make themrecognizable.

3. Something the user *is*a These authenticators, called biometrics, are based on a physical characteristic of the user, such as a fingerprint, the pattern of a person's voice, or a face (picture). These authentication methods are old (we recognize friends in person by their faces or on a telephone by their voices) but are just starting to be used in computer authentication.

Passwords asAuthenticators

The most common authentication mechanism for user to operating system is a password, a"word" known to computer and user. Although password protection seems to offer a relatively secure system, human practice sometimes degrades its quality. In this section we consider passwords, criteria for selecting them, and ways of using them for authentication. We conclude by noting other authentication techniques and by studying problems in the authentication process, notably Trojan horses masquerading as the computer authentication process.

Use of Passwords

Passwords are mutually agreed-upon code words, assumed to be known only to the user and the system. In some cases a user chooses passwords; in other cases the system assigns them. The length and format of the password also vary from one system to another.

Even though they are widely used, passwords suffer from some difficulties of use:

- Loss. Depending on how the passwords are implemented, it is possible that no one will be able to replace a lost or forgotten password. The operators or system administrators can certainly intervene and unprotect or assign a particular password, but often they cannot determine what password a user has chosen; if the user loses the password, a new one must beassigned.
- Use. Supplying a password for each access to a file can be inconvenient and time consuming.
- Disclosure. If a password is disclosed to an unauthorized individual, the file becomes immediately accessible. If the user then changes the password to reprotect the file, all other legitimate users must be informed of the new password because theirold
  password will fail.
- Revocation. To revoke one user's access right to a file, someone must change the password, thereby causing the same problems asdisclosure.

The use of passwords is fairly straightforward. A user enters some piece of identification, such as a name or an assigned user ID; this identification can be available to the public or easy to guess because it does not provide the real security of the system. The system then requestsa password from the user. If the password matches that on file for the user, the user is authenticated and allowed access to the system. If the password match fails, the system requests the password again, in case the usermistyped.

VI. DESIGNING TRUSTED O.S

Operating systems are the prime providers of security in computing systems. They support many programming capabilities, permit multiprogramming and sharing of resources, and enforce restrictions on program and user behavior. Because they have such power, operating systems are also targets for attack, because breaking through the defences of an operating system gives access to the secrets of computing systems.

In we considered operating systems from the perspective of users, asking what primitive security services general operating systems provide. We studied these four services:

1.memory protection
2. fileprotection
3. general object access control 4.userauthentication

We say that an operating system is trusted if we have confidence that it provides these four services consistently and effectively. In this paper, we take the designer's perspective, viewing a trusted operating system in terms of the design and function of components that provide security services. The four major underpinnings of a trusted operating system:

1. Policy. Every system can be described by its requirements: statements of what thesystem should do and how it should do it. An operating system's security requirements are a set of well-defined, consistent, and implementable rules that have been clearly and unambiguously expressed. If the operating system is implemented to meet these requirements, it meets the user's expectations. To ensure that the requirements are clear, consistent, and effective, the operating system usually follows a stated security policy: a set of rules that lay out what is to be secured and why.

ISSN: 2319 – 8753

International Journal of Innovative Research in Science, Engineering and Technology
Vol. 1, Issue 1, November 2012

2. Model. To create a trusted operating system, the designers must be confident that the proposed system will meet its requirements while protecting appropriate objects and relationships. They usually begin by constructing a model of the environment to be secured. The model is actually a representation of the policy the operating system will enforce. Designers compare the model with the system requirements to make sure that the overall system functions are not compromised or degraded by the security needs. Then, they study different ways of enforcing that security.

3. Design. After having selected a security model, designers choose a means toimplement
it. Thus, the design involves both what the trusted operating system is (that is, itsintended functionality) and how it is to be constructed (its implementation).

4. Trust. Because the operating system plays a central role in enforcing security, we (as developers and users) seek some basis (assurance) for believing that it will meet our expectations. Our trust in the system is rooted in two aspects: features (the operating system has all the necessary functionality needed to enforce the expected security policy) and assurance (the operating system has been implemented in such a way that we have confidence it will enforce the security policy correctly and effectively).

VII. SECURITY POLICIES

To know that an operating system maintains the security we expect, we must be able tostate its security policy. A security policy is a statement of the security we expect the system to enforce. An operating system (or any other piece of a trusted system) can be trusted only in relation to its security policy; that is, to the security needs the system is expected to satisfy.

Military Security Policy

Military security policy is based on protecting classified information. Each piece ofinformation is ranked at a particular sensitivity level, such as *unclassified*, *restricted*, *confidential*, *secret*, or *top secret*. The ranks or levels form a hierarchy, and they reflect an increasing order of sensitivity.

Commercial Security Policies

Commercial enterprises have significant security concerns. They worry that industrial espionage will reveal information to competitors about new products under development. Likewise, corporations are often eager to protect information about the details of corporate finance. So even though the commercial world is usually less rigidly and less hierarchically
structured than the military world, we still find many of the same concepts in commercial security policies. For example, a large organization, such as a corporation or a university, maybe divided into groups or departments, each responsible for a number of disjoint projects. There may also be some corporate-level responsibilities, such as accounting and personnel activities. Data items at any level may have different degrees of sensitivity, such as *public*, *proprietary*, or *internal*; here, the names may vary among organizations, and no universal hierarchy applies.

VIII. MODELS OF SECURITY

In security and elsewhere, models are often used to describe, study, or analyze a particular situation or relationship. McLean gives a good overview of models for security. In particular, security models are used to test a particular policy for completeness and consistency document a policy help conceptualize and design animplementation check whether an implementation meets its requirements.

We assume that some access control policy dictates whether a given user can access a particular object. We also assume that this policy is established outside any model. That is, a policy decision determines whether a specific user should have access to a specific object; the model is only a mechanism that enforces that policy. Thus, we begin studying models by considering simple ways to control access by one user.

Multilevel Security

Ideally, we want to build a model to represent a range of sensitivities and to reflect the need to separate subjects rigorously from objects to which they should not have access. For instance, consider an election and the sensitivity of data involved in the voting process. Thenames of the candidates are probably not sensitive. If the results have not yet been released,
the name of the winner is somewhat sensitive. If one candidate received an embarrassingly low number of votes, the

vote count may be more sensitive. Finally, the way a particular individual voted is extremely sensitive. Users can also be ranked by the degree of sensitivity of information to which they can have access. For obvious reasons, the military has developed extensive procedures for securing information.
The generalized model is called the lattice model of security because its elements form a mathematical structure called a lattice. In this section, we describe the military example and then use it to explain the lattice model.

IX. CONCLUSION

A generalization of the military model of information security has also been adopted as a model of data security within an operating system. In 2005, Bell [BEL05] returned to the original model to highlight its contribution to computer security. He observed that the model demonstrated the need to understand security requirements before beginning system design, build security into not onto the system, develop a security toolbox, and design the system to protect itself.

REFERENCES

1. Sarfraz Alam, Mohammad M. R. Chowdhury, Josef Noll, 2010 SenaaS:An Event-driven Sensor Virtualization Approach for Internet of Things Cloud, NetworkedEmbedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on,1-6.
2. Clark, C. (2009). Cloud computing and mobility: Adjusting in a New World – From Threat to Ally. Retrieved from http://www.europeanbusinessreview.com/?p=5152
3. ENISA, C. C. (2009). Benefits, risks and recommendations for information security. European Network and Information Security.
4. Friedman, A. A., & West, D. M. (2010). Privacy and security in cloud computing. Center for Technology Innovation at Brookings.
5. Ganore, P. (2010). Cloud Computing and its Advantages. Retrieved from http://www.esds.co.in/blog/cloud-computing-and-its-advantages/
6. Daoliang Li, Yingyi Chen, Oct. 2010, Computer and Computing Technologies in Agriculture. Springer,24-31.