Protection and Security Issues for Future Systems

B. Clifford Neuman

Department of Computer Science and Engineering
University of Washington

1 Introduction

We are becoming increasingly dependent on computers in daily life. This dependence brings with it a heightened need for security in the computer systems we use. The distributed nature of recent systems has made it difficult to apply many of the security techniques used in centralized systems. Additionally, many of the services which are becoming available by computer are placing new demands on the protection and security mechanisms of the systems on which they run. These services require interaction between parties that are mutually suspicious of one another; the servers require protection from users, while at the same time the users require protection from malicious or incompetent service providers.

This paper examines the problems of protection and security as applied to future computer systems. It begins in Section 2 with a discussion of the goals of protection and security and with an enumeration of the aspects of the problems. Section 3 describes the techniques used to implement security in existing systems. Section 4 starts by examining the relationship between centralized and distributed systems, considering future

The author's research was supported in part by the National Science Foundation (Grant No. CCR-8619663), the Washington Technology Centers, and Digital Equipment Corporation.

Author's address: University of Southern California, Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, California 90292. U.S.A. (bcn@isi.edu)

Presented: Workshop on Operating Systems of the 90s and Beyond.
Dagstuhl Castle, Germany. July 1991.
Springer-Verlag Lecture Notes in Computer Science #563 pages 184-201.

directions in computer systems, and highlighting the protection and security issues that will grow in importance. It then describes recent work that addresses some of the issues and discusses possible solutions for others. Section 5 looks at the problem of privacy and how it is related to security. Policies that could improve privacy in future systems are discussed, and where appropriate, mechanisms are suggested to implement those policies. The paper concludes with a brief summary of the issues.

2 Aspects of Protection and Security

Protection and security have been important issues in the design of computer systems since the introduction of multiprogramming. Originally the goal of protection was to prevent one running program from interfering with another. Once computers stored information belonging to more than one user it became important to restrict access to information and resources so that they could only be accessed by the appropriate users.

There are many aspects to protection and security in computer systems. *Protection* prevents a process from accessing a resource that it shouldn't. The protection of resources depends on *authorization*, which specifies the resources that are to be accessible to a user or process. *Authentication* involves verifying the identity of a user or process so that it can be decided which resources may be accessed. Most authorization mechanisms depend on the ability of the system or service to authenticate the user.

The *security* of a system is a measure of its ability to implement a security policy. Security depends on the integrity of the protection mechanism, on its being flexible enough to implement the desired security policy, and on its proper use.

In addition to the above, accounting and audit are important security tools. *Accounting* mechanisms maintain a record of the resources used, facilitate charging for the use of resources, and enforce the fair allocation of limited resources across a community of users.

Audit serves two functions. The first is *assurance*: certifying that the a system meets certain security requirements. This involves verification of the software, and checking that it has been installed and configured in a manner that implements the desired security policy. The second function is to monitor a system's operation (through logs and other techniques), to identify security breaches and to facilitate corrective action.

A final issue of importance is privacy. *Privacy* is concerned with the ability of users to control the release of information that they own or that concerns them. The privacy of a system depends in part on the ability of users to protect information from the administrators or owners of the systems or services that store it.

3 Protection and Security in Existing Systems

This section describes the techniques that implement security in existing systems. It starts with a discussion of security mechanisms used in centralized systems and then shows how some of these mechanism have been extended to work in distributed systems. There is much that could be written about protection and security in existing systems, more than can be covered in a single paper. This paper only describes past work to the extent necessary to lay the groundwork for discussion of the future. The reader is referred to [12] for a more complete description of protection in existing systems.

3.1 Centralized Systems

In centralized systems, protection and security depend on the ability of a trusted kernel to control access to protected resources and on the existence of a secure communication channel between the user (or user's process, called the client) and the kernel. In most centralized systems, these requirements are met by the computer hardware; restricted operations such as input and output can be performed only by the kernel. When the client needs to perform such operations it calls procedures in the kernel, but only starting from certain entry points. The kernel performs the operation on the client's behalf after first checking to make sure that the client is allowed to perform the requested operation. Although memory access does not typically require intervention by the kernel, the kernel must first grant permission for a process to access memory by setting up a table used by the hardware to control access.

In order to enforce a security policy, the kernel requires a method to represent the policy. Conceptually, the policy may be represented by an access matrix [5], the rows of which represent protection domains (the sets of privileges available to individual users or processes at a particular time) and the columns of which represent objects. Each entry in the matrix lists the rights of a process executing in a particular protection domain to access an object.

Within a system, the access matrix can be represented in two ways: by an access control lists or by capabilities. An *access control list* (ACL) can be associated with each object listing the individuals (principals) authorized to access the object along with the access rights authorized (e.g., read, write, etc). Alternatively, each process maintains a list of capabilities. A *capability* identifies an object and specifies the access rights authorized to the process holding the capability. It must not be possible for a process to directly modify its capability list. If it could, the process could add capabilities to objects for which it was not authorized access. In some systems capabilities can be passed from process to process allowing a process to grant rights to others. The transfer is mediated by the operating system kernel which enforces policy restrictions on such transfers.

There are tradeoffs between the two approaches regarding the ease of checking access versus the ability of the owner of an object to directly control or determine who is allowed access. There are also tradeoffs between the ease of propagating access rights from one process to another versus the ability to revoke access should that become necessary. Both approaches depend on the ability of the system to identify its users. The name of the user on whose behalf a process is running must be known to check an access control list. Although the name of a user is not needed to check a capability, the system must know the user's name in order to provide the correct set of initial capabilities to the user's process.

Most centralized systems authenticate the user through the use of a password. The user presents a password to the kernel which checks to make sure the password is correct. If correct, the user's identity is associated with the running process and/or the appropriate capabilities are made available.

Accounting and audit in centralized systems depend on the ability of the kernel to maintain a log of the actions taken by users, to keep track of the resources used, and to maintain quotas.

Much has been accomplished in the implementation of protection and security mechanisms for centralized systems. Many of those mechanisms depend on the ability of the kernel to control access to protected resources and on the existence of secure communication between the user and the kernel. The next section discusses how to implement such mechanisms when those assumptions no longer hold.

3.2 Distributed Systems

Control of access to resources in distributed systems occurs at a higher level than in centralized systems. Before passing through the kernel, control must first pass through the server maintaining the object (often referred to as the end-server). It is up to the kernel to make sure that only the end-server can directly access the objects it maintains. It is up to the end-server to accept requests, to decide if the request is compatible with the applicable security policy, and if so to perform the operation.

The same mechanisms used by the kernel can be used by the end-server to make access control decisions. The mechanisms must be extended to work in an environment where the client might run on a system that is not trusted by the server or where communication between the client and server is over an insecure channel (a network).

For example, if a server relies on an access control list it must first accurately identify the client making a request. A simple password scheme will no longer work since an attacker could eavesdrop on the network, steal the password, and use the stolen password to impersonate the client. Authentication in distributed systems is accomplished using encryption-based protocols such as those described by Needham and Schroeder [8] or implemented in Kerberos [14].

Similarly, if authorization is based on capabilities, the server cannot rely on the client's system to prevent the client from changing the capabilities it holds. This is solved in Amoeba [15] by cryptographically sealing the capabilities. Even cryptographically sealed capabilities, however, must be protected when sent across the network, including when they are presented to the server. Without such protection an eavesdropper could steal the capability and make use of it itself.

The reader is referred to [17] for a more detailed discussion of the threats that arise from communication through computer networks. There has been more work on distributed systems security than has been described so far. Much of it has not yet made it into actual systems. Some aspects of this work will be important for future systems and will be described later in the paper.

4 The Future of Computer Security

Future operating systems will rely heavily on distributed services available from the network. This section starts by discussing the differences between the protection mechanisms employed by centralized and distributed systems. It then looks at the services which will be available in future systems, highlights the protection and security issues that will be important for these services, and describes recent work aimed at addressing some of those issues.

In centralized systems, protection and security reside in the kernel. In distributed systems, they reside in the server. Resources can only be accessed through the interface specified by the server, thus preventing the client from circumventing the protection implemented by the server. Servers are not complete systems; they may themselves rely on other servers, either to perform part of the requested operation, to verify the identity of the client, or to perform other related operations.

As the variety of services available on computer networks increases, the frequency with which one accesses services provided outside one's own organization will also increase. The computer network will become an extension of the marketplace. Most servers will be unable to make authorization decisions in isolation. Instead, they will depend on other services to help them make those decisions. Some services will charge clients for the service provided. These services might make use of a distributed accounting service to obtain payment for the services they provide.

In future systems, users will be able to choose from a large number of service providers. Users will also be able to run large numbers of programs on their own systems, and these programs might be retrieved over the network. Clients will demand guarantees about the security of the services they use and the programs they run. Such guarantees might require the use of electronic signatures, thus placing new demands on authentication mechanisms. Finally, clients might demand notification from servers when certain actions are taken that affect their security or privacy.

4.1 Protection

Protection is provided in distributed systems by **both** the kernel **and** the server. When other processes run on the same system as a server, this protects the other processes from the server's errors. If a system runs only a single server, or if the server is running code whose security has been verified, some of the kernel checks will be unnecessary. Future systems might allow verified code to run in a mode that supports direct access to the system's resources (bypassing the kernel). This will improve performance by eliminating redundant checks and by reducing the number of context switches. Such an approach does not eliminate the kernel checks, instead it allows the checks to be performed in advance at the time the code is verified. This approach is being used by OCCAM where verification is performed by the compiler.

4.2 Authorization

Authorization in most distributed systems is left to the end-server. The server maintains an access control list identifying those clients authorized to use the service or to access a particular object maintained by the server. Because of the number of potential clients it will become less practical for each server to maintain its authorization database in isolation from others. Instead, authorization decisions will need to be based on information local to the server together with authorization information provided by authorization, group, and accounting services.

Authorization mechanisms should allow a client authorized to perform an operation to allow another principal to perform that operation on the client's behalf. This is important when a client uses a server that must call another server to perform part of the operation. Such delegation can be supported through passports [13] or within the Digital Distributed System Security Architecture [4]. Delegation in these systems is an authentication issue. The intermediate server assumes the identity of the client. In the DSSA, if only a subset of the client's rights are to be made available to the intermediary, a separate principal must be created with restricted rights. This principal is called a role, and the intermediary assumes the limited rights of the role. With passports, restrictions can be added to the passport itself.

Restricted proxies [9] provide an alternative mechanism designed specifically for authorization. Restricted proxies support a broad range of authorization models including authorization services, group services, and capabilities. In fact, restricted proxies are similar to capabilities in that the end-server can verify the authenticity of the proxy on its own. It differs from capabilities because presentation of a proxy might require the client to take part in a cryptographic exchange with the end-server to prove its possession of the accompanying encryption key. Restricted proxies are more general than the other mechanisms described and they allow a combination of authorization models to be employed by the end-server.

4.3 Accounting

Secure and distributed mechanisms for accounting will play an important role in the introduction of new services available through computer networks. Some services will charge clients for the service they provide. Alternatively, cooperating groups of servers might impose quotas on the total resources available to a client, and those quotas will have to be maintained across the cooperating servers. As the use of servers outside one's own organization increase, the need for secure distributed accounting mechanisms will become critical.

Among the mechanisms that have been proposed for distributed accounting are Amoeba's bank server and restricted proxies. In Amoeba, a client contacts a bank server to transfer funds into the end-server's account before it requests service [7]. The end-server then provides service until the pre-paid funds have been exhausted.

Restricted proxies [9] support two models for distributed accounting. The first is analogous to a certified check. The client first obtains authorization in the form of a restricted proxy from an accounting server. The proxy is forwarded to the end-server and the operation is performed. The end-server then forwards the check to the accounting server and the funds are transferred. The second mechanism is analogous to the personal check. The client presents the end-server with a restricted proxy allowing the end-server to transfer funds from the client's account. If the client and the end-server use different accounting servers, the end-server can issue an endorsement authorizing its accounting server to collect the funds on its behalf.

Both mechanisms support multiple currencies thus allowing the accounting servers to maintain cash balances as well as quotas in various units. Quotas are implemented by withdrawing funds as a resource is allocated and by depositing funds when the resource is freed.

4.4 Authentication Revisited

Future systems will place new requirements on authentication. They will make use of digital signatures to verify the authenticity of programs, messages, and documents. By basing such signatures on public key cryptography [3], the signatures can be verified by multiple users. Among the authentication systems under development that are based on public-key cryptography are SPX [16] and X.509 [2].

While public key cryptography is the appropriate choice for digital signatures, its advantage over conventional cryptography for one-to-one authentication is less clear. The primary advantage of using public key techniques for one-to-one authentication is that the certification authority can be kept offline, decreasing its vulnerability to attack (the certification authority certifies that a key belongs to a particular principal). Unfortunately, using an offline certification authority requires the use of long-lived credentials and such credentials are difficult to revoke if a key is compromised. Mechanisms that can be used to support revocation reintroduce vulnerability at the authentication server.

The main cost of basing authentication on public key cryptography is that it requires additional assumptions about the properties of the cryptosystem. These assumptions significantly reduce one's options in choosing the cryptosystem. In fact, at the present time, RSA [11] is the only practical choice for a public key cryptosystem.

The primary vulnerability of existing authentication techniques is not the certification authority. It is the user's choice of passwords. If encryption keys are derived from passwords, and if those passwords are poorly chosen, an attacker can try every word in the dictionary in an attempt to guess the password. Existing encryption-based authentication techniques allow this attack to occur without significant interaction with the authentication server. The attacker repeatedly decrypts text encrypted in the users key, and checks whether it yields the expected result. Lomas et al. [6] have shown how an authentication protocol can be designed which does not give the attacker the information needed to verify the correctness of the guess.

The second point of vulnerability for existing authentication mechanisms is the computer used by the client. Unless the computer is always under the client's control, there is little guarantee that the software has not been modified to steal the client's password. The effect of such an attack can be reduced with the introduction of smartcards capable of performing the encryption necessary to take part in the authentication protocol without divulging the client's key to the workstation [1]. An alternative is the use of one-time smartcard-generated passwords in conjunction with the authentication protocol. Work is underway to add such functionality to Kerberos. Such an approach can limit the damage that can result from using an untrusted system, but it depends on the security of the algorithm that generates the one-time passwords.

4.5 Assurance and Audit

Future systems will support a large number of service providers and there will be many sources from which users can obtain programs. Mechanisms will be necessary to assure users that a service provider is trustworthy, or that a program is both safe and hasn't been altered. In society there are several mechanisms which provide these assurances.

Bonding, licensing, and professional society membership provide assurance that a service provider is qualified to perform a service and that it has not been found to be untrustworthy. In computer systems, these assurances can be represented by certificates digitally signed by the licensing authority, by the professional society, or by the agency issuing the bond. These credentials would be granted to a server after it has met the requirements set forth by the organization issuing the credentials.

In the U.S., products meeting certain safety requirements are approved by the Underwriters Laboratories. In computers, programs that have been reviewed and found to meet certain security criteria could receive similar approval. Such approval can be communicated through a digitally signed certificate that includes a checksum of the reviewed program. The checksum is analogous to a tamper-evident seal allowing the detection of subsequent changes. Such techniques have been considered as a way to limit the spread of viruses [10].

The mechanisms described above will require the user to decide which authorities it trusts to provide assurances about the security of the programs and servers it uses. Such assurances could be automatically checked before a program is run, or when a server is contacted. Where such assurances are not present, the client might still use the program or server, but might restrict the privileges that are available or delegated to it.

The guarantees described so far are designed to improves one's confidence in the security of a server or program before it is used. Users also want assurances of the continued security of a service. Audit trails are important for verifying that the security policy has been correctly implemented, or for taking action when a breach has occurred. In a centralized system, the audit trail can be maintained by logging events that affect the security of the system. In future systems, however, a client entrusting information to a server might be in a better position to determine if an event is significant. Future systems should allow clients that entrust information to servers to specify the types of events for which notification will be provided.

Notifying users about events that concern them will require a secure mechanism for notification. Such a mechanism should guarantee that a message sent will eventually be delivered. Upon delivery, a message might be brought to the user's attention or recorded in the user's personal log file with a suitable expiration date. In addition to its use for notifying users of security related events, a notification service might be used to provide assurance that an expected event has occurred, for example, receipt of a mail message.

5 Privacy

The discussion so far has been focused on computer security. Let me now discuss the issue of privacy, the ability of users to control the accessibility of information about themselves. Privacy and security interact in many ways. If information about an individual has been collected for a particular purpose, security must be employed to prevent others from gaining access to that information. Security also protects the privacy of the user's own files, but complete privacy requires that the files also be protected from the maintainers of the system on which they are stored. Finally, logs of a user's actions affect the user's privacy, yet they help maintain the security of the system and they aid in the detection of breaches of the privacy of others.

Future systems should protect information by default. If information is to be made available to others it should require specific action by the owner of the information. Additionally, the mechanism used to specify protection information for files must be flexible enough to truly reflect the intended access policy of the owner. Unless both conditions are met, it is difficult to determine if the readability of a file is intentional or an oversight. Once met, inclusion in the access control list for a file either individually, as a member of a group, or because the file is world readable, can be considered sufficient consent to access the file unless additional factors indicate otherwise.

A problem to be addressed in future systems is that information is often retained longer than necessary. This applies to information maintained about individuals, as well as information maintained by the individuals themselves. Future systems should associate an online expiration date with the information they store. After that date, the information should be deleted. A default expiration date suitably far into the future should be applied to most files. Seventy-five years might be appropriate, though a perpetual life should be assigned for any material that has been identified as having historical or archival significance. Information which has a limited useful (or legal) life for its intended purpose should expire when its useful life has elapsed.

It should be possible to extend the life of information that is about to expire, but doing so should require specific action. Similarly, it should be possible to restore expired information, but only for a limited period after its expiration.

Users are often not aware of the potential affect backup policies have on their privacy. A case in point was the retrieval of old mail messages from backup tapes during the investigation of the Iran-Contra affair. The problem is worse when backups are available online. Some systems automatically back up all files that change in the course of a day to optical disk. Old and deleted versions can be referenced by including the date on which they existed as part of the file name. Unfortunately, if a user discovers that the protection of a file was incorrectly set, the change in protection affects only the current copy, not the backup copies. A similar problem arises when the user subsequently encrypts the file. Backup copies of the unencrypted version might still be available.

Just as information should have an online expiration date, a separate limit should apply to the retention period for backup copies. This date should typically be shorter than the online expiration as it limits the period that data may continue to exist after it has been deleted by the user. It should be possible to set this limit on a per file basis. Doing so may require changes to backup procedures in order to accommodate such limits, but such changes are important to protect the user's privacy. Additionally, if backups are available online, it should be possible to make the backup copies of a particular file inaccessible.

The setting of expiration dates is a matter of policy. Thus, government agencies might require that copies be maintained of all correspondence. The policy needs to be clear to the user and associating explicit expiration dates with information does that.

It should be possible for users to protect information from system maintainers or service providers. This not only protects the privacy of individuals against inquisitive system maintainers, but it provides an additional barrier for attackers that are able to breach the security of the system. Such protection might be provided by supporting the automatic encryption of data before it is written to disk or sent to a server. The key under which the data is encrypted can be stored with the file itself, but encrypted under the higher level keys of the principals authorized to access the file. Services can be created which manage the keys and which issue the keys after authenticating a client and verifying that the client is authorized to access the file. Such services would allow the access decision for the information in the file to be handled by a separate server that might be more secure than the server storing the data itself. In fact, such a key server might be simple enough that it might be implemented on a smartcard.

Finally, careful consideration should be given to the information maintained in logfiles. Information identifying clients and their actions might be included for the purpose of identifying and correcting security breaches. Where such identifying information is included, care must be taken to prevent its disclosure to others and to limit its use to the intended purposes. Additionally, such identifying information should be eliminated from the log as soon as practical. When information identifying an individual will be retained for any significant time (e.g., longer than one week), that individual should be notified of the nature of the information and the length of time it will be retained. Such notification might be supported by the same mechanism required to notify clients of events affecting information entrusted to servers.

There are tradeoffs to be made between the protection of the privacy of individual users and the security of the system as a whole. Where the balance lies is a decision to be made by the maintainers and users of future systems and services. It is hoped that the design of future systems will take these issues into account and that the interfaces to such systems will allow users to specify the measures to be taken to protect their privacy. Where a client and a server cannot agree on the measures to be taken, the client can choose a different server. In the absence of a court order directing otherwise, a server accepting a request should be required to make its best efforts to satisfy the privacy constraints specified in the request and to notify the client if it is unable to do so.

6 Conclusions

In the coming years an increasing number of services will become available by computer. Many of these services will require interaction between parties that do not completely trust one another. Computer networks and the systems connected to them will, in effect, become an electronic marketplace. The users of such systems will demand protection from the actions of others.

Many of these protections can be provided by computer-based mechanisms analogous to the mechanisms that protect individuals in society. Among the protections to be supported are authentication, authorization, accounting, assurance, and audit. Some of the mechanisms that provide these protections have been described in this paper. The future of these mechanism will depend on building the infrastructure that will support their widespread use.

The growing use of computers in daily life has the potential to increase the accessibility of information that individuals might prefer to keep private. Future computer systems should incorporate mechanisms that support the preservation of individual privacy.

Acknowledgments

I would like to thank Terry Gray, David Keppel, Ed Lazowska, Hank Levy, Jennifer Steiner, and Donald Tsang who commented on earlier drafts of this paper.

References

[1] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and delegation with smart-cards. Technical Report 67, Digital Equipment Corporation Systems Research Center, November 1990.

[2] CCITT. Recommendation X.509: The directory authentication framework, December 1988.

[3] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[4] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson. The Digital distributed system security architecture. In *Proceedings of the 1989 National Computer Security Conference*, pages 305–319, 1989.

[5] B. W. Lampson. Protection. In *Proceedings of the 5th Princeton Symposium on Information Sciences and Systems*, March 1971.

[6] T. Mark Lomas, Li Gong, Jerome H. Saltzer, and Roger M. Needham. Reducing risks from poorly chosen keys. In *Proceedings of the 12th ACM Symposium on Operating Systems Principles*, pages 15–18, December 1989.

[7] S. J. Mullender. *Principles of Distributed Operating System Design*. PhD thesis, Vrije Universiteit, Amsterdam, October 1985.

[8] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, December 1978.

[9] B. Clifford Neuman. Proxy-based authorization and accounting for distributed systems. Technical Report 91-02-01, Department of Computer Science and Engineering, University of Washington, March 1991.

[10] M. M. Pozzo and T. E. Gray. A model for the containment of computer viruses. In *Proceedings of the 2nd Aerospace Computer Security Conference*, pages 11–18, December 1986.

[11] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[12] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), September 1975.

[13] Karen R. Sollins. Cascaded authentication. In *Proceedings of the 1988 IEEE Symposium on Research in Security and Privacy*, pages 156–163, April 1988.

[14] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Winter 1988 Usenix Conference*, pages 191–201, February 1988. Dallas, Texas.

[15] Andrew S. Tanenbaum, Robbert van Renesse, Hans van Staveren, Gregory J. Sharp, Sape J. Mullender, Jack Jansen, and Guido van Rossum. Experience with the Amoeba distributed operating system. *Communications of the ACM*, 33(12):47–63, December 1990.

[16] J. Tardo and K. Alagappan. SPX: Global authentication using public key certificates. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1991.

[17] V. L. Voydock and S. T. Kent. Security mechanisms in high level network protocols. *ACM Computing Surveys*, 15(2), June 1983.