EDPACS

THE EDP AUDIT, CONTROL, AND SECURITY NEWSLETTER

JUNE 1996 VOL. XXIII, NO. 12

SECURITY CONTROLS FOR A CLIENT/SERVER ENVIRONMENT

JENNIFER BAYUK

Client/server is a common term for a generic computing architecture in which desktop software uses shared information processing resources that are not located actually on the desktop. The desktop software is the client. The computing platform that holds the shared resources is the server.

This architecture is implemented in many different ways. There is no line of demarcation that separates definitively all of the client/server applications from other information processing applications that involve communication between desktop software and some number of servers. However, a simple terminal emulation package generally is held not to be an example of a client/server arrangement. But an application-builder software product that functions as a front-end to multiple UNIX data bases generally is held to be an example of a client/server mechanism.

The loose definition of a client/server application gives systems engineers broad latitude with which to design specifications for a client/server information architecture. This article:

- ☐ Presents the options that systems engineers have in specifying a client/server architecture.
- ☐ Explains the security vulnerabilities of these architecture options.
- ☐ Demonstrates how the particular architecture choices that the systems engineer makes constrain the choice of the client/server security controls. (Specifically, this article describes how the architectural implementation of user access and client-to-server connectivity prescribes the type of control that is effective in providing assurance that the client/server information is secure.)
- ☐ Offers the information systems auditor guidance in getting started in the client/server controls testing process.

THE CLIENT/SERVER ARCHITECTURE OPTIONS

The client/server architecture access options are:

- ☐ Host-based.
- ☐ Application-based.
- ☐ Data base management system (DBMS)–based.

Editor
BELDEN MENKUS, CISA

AUERBACH PUBLICATIONS
A Division of
Warren, Gorham & Lamont

A SIMPLE TERMINAL EMULATION PACKAGE GENERALLY IS HELD NOT TO BE AN EXAMPLE OF A CLIENT/SERVER ARRANGEMENT.

Host-based access employs such authentication mechanisms as the file transfer protocol of the host operating system control access. Application-based access means that a particular application has been written that runs as one user on the server and that all of the clients have access to this application by virtue of their being authenticated to the application process. DBMS-based access means that the authentication mechanisms are provided by a server-located mechanism (e.g., one that employs an array of third-party reporting tools). Any of these three types of access mechanisms are automated from the desktop.

The client/server architecture connectivity options include:

- ☐ A direct private line to a server.
- ☐ A LAN to a server.
- ☐ A LAN/WAN combination to a server.

In the direct private line scenario, the server port still is running under the legacy system protocol. The communication mechanism that is being employed operates in a serial mode. It even may be synchronous. The line admits one connection at a time. In the LAN or WAN scenario, the server port has been converted to the protocol of the LAN. It becomes a node on the network and receives the same network traffic as does any workstation. The tendency is for a progressively wider range of resource users to have access to any particular server. This allows a particular piece of client software to be deployed anywhere on the network. Where the user connectivity is accomplished through a LAN or a WAN, the controls that are employed must assume that the client/server data potentially is available at every access point on the network.

THE CLIENT/SERVER SECURITY ISSUES

The information systems security threats to a client/server architecture are:

- ☐ Spoofing.
- ☐ Browsing.
- ☐ Penetration.
- ☐ Eavesdropping.

Spoofing involves posing as a legitimate network access point. This might result from configuring several TCP/IP network services to use the host name of a machine to authenticate legitimate users. However, an unauthorized user can change the host name of his or her own computer to the name of an authorized host and then spoof the targeted TCP/IP service into authorizing some number of otherwise unauthorized users.

Browsing involves an attempt by a user to access services for which this person has no authentication. The services may be ones for which no authentication is required (i.e., the world wide web). In addition, the services may be those that require a password, such as the log-in process.

Penetration involves bypassing the intended user information processing application to be able to attack the server itself. The penetrator may then access other applications on the server, disrupt some or all of the applications on the server, or install computer viruses or Trojan horses on the server itself.

Eavesdropping involves capturing data from a shared communications vehicle (i.e.,a wide area network). The data that is acquired may include a log-in sequence, revealing a log-in name and its corresponding password. Hence, eavesdropping may be used to facilitate browsing.

The client/server architecture connectivity mechanisms are used determine the type of security threat that may occur. Direct connectivity, thus, is subject to browsing. LAN or WAN connectivity, however, is subject to both eavesdropping and browsing. Access mechanisms, then, are relied on to control the threats that may be introduced by employing a particular connectivity architecture. However, these mechanisms in turn are subject to threats.

Desktop software is dispersed widely and fully. It is under the control of a diverse population of users who employ ever more powerful machines. The only safe assumption is to not rely in any way on a secure desktop configuration. No passwords or other authentication mechanisms should be built into the desktop client/server application software. Rather, reliance for both local and remote access controls must be placed on the server itself. But access to the server is subject to both spoofing and penetration.

The information systems security architecture has the same components regardless of configuration of the computing architecture that is to be secured. Despite the inherent flexibility of these configurations, the client/server security controls follow the same principles as did their predecessors:

- ☐ Prevention (also known as access control).
- ☐ Detection.
- ☐ Recovery.

A server must control against the possible corruption of its clients by being unspoofable. It must employ mechanisms that allow for the authentication of users, rather than the authentication of machines. In addition, the server must be unpenetrable. It should allow access only to those resources that are required by the specific application. However, if a server is spoofed or penetrated, it should detect the fact that this has occurred. Even if the server fails to detect that it has been compromised, it must be able to recover from a successful attack.

HOST-BASED ACCESS EMPLOYS SUCH AUTHENTICATION MECHANISMS AS THE FILE TRANSFER PROTOCOL OF THE HOST OPERATING SYSTEM CONTROL ACCESS.

CLIENT/SERVER PREVENTION CONTROLS

For each combination of access and connectivity architecture, unique threats and corresponding controls exist that should be applied to counter them.

Controls for Host-Based Access

The most effective server host-based access control for protecting against the occurrence of spoofing and browsing is to disable all of the host-provided services that are not absolutely necessary for client applications. Thus, there should be no developer emergency access in place, no use of production machines for handling routine electronic mail, and no file transfer mechanisms enabled to facilitate biannual software updates. These

convenient extras introduce security vulnerabilities that even yet may be undiscovered. By minimizing the number of services that are available on a given server, an administrator minimizes the complexity of access administration.

Host-based access controls against penetration include employing restricted menus and narrowly defined user groups. Restricted menus ensure that users may not wander from their intended path into other areas of the operating system. These may not be within the focus of the existing security efforts. These controls include application scripts that start to function when a client is connected and that disconnect the client connection when the script is exited. (The exit may be controlled by the application or it may be the result of an unintended event (e.g.,the user pressing a break key.) Narrowly defining both the user groups and the appropriate file permission configuration will ensure that even if users do wander they cannot get far enough to penetrate vital host services.

BROWSING INVOLVES AN ATTEMPT BY A USER TO ACCESS SERVICES FOR WHICH THIS PERSON HAS NO AUTHENTICATION.

Host-based access controls against eavesdropping are encryption and token-based authentication. The degree to which the information in client/server arrangements should be encrypted depends on the confidentiality of this information. Log-in sequences are the primary examples of highly confidential information. If eavesdropping reveals a particular log-in and password combination, the corresponding account will be compromised. Where both the log-in sequence and the information that is transmitted between the client and the server are confidential, the only way to control eavesdropping is through encryption.

Encryption may be implemented in the hardware communication device, the software communications protocol, or in the information processing application software. If encryption is implemented in the application software, then it is possible to restrict encryption only to the most sensitive data items. The items that are protected always should include the log-in sequence. (This restriction may be enforced for performance reasons.) However, if encryption is done in the hardware device or in the communications protocol, then the same mechanism automatically will encrypt the log-in sequence as well.

When only the log-in sequence is confidential, another way to control eavesdropping compromise of the log-in sequence is to make the captured information useless by employing authentication mechanisms that require the use of hand-held tokens. These devices replace the conventional password with an authentication message that changes with each access. This message change prevents the reuse of an eavesdropped authentication message for unauthorized access.

Controls for Application-Based Access

Application-based access controls for protecting against the occurrence of spoofing and browsing are those application subroutines that comprise the access mechanisms themselves. No assumptions should be made concerning the following:

☐ Robustness of these mechanisms.

- ☐ Degree to which encryption is used in storing or transmitting passwords.
- ☐ Features of these mechanisms being coincident with those available from the underlying host operating system or DBMS.

Application-based access mechanisms are written entirely independently of either the host operating system or the DBMS. Usually this is done to facilitate the portability of the source code. Hence, these mechanisms do not use the authentication and authorization mechanisms that are provided by the host operating system or the DBMS. Authorization is granted to execute an application subroutine rather than to view or to modify a data file. In addition, the application user or administrator need not be a privileged UNIX or DBMS user.

Because every application has a different method of implementing the access mechanisms that it uses, careful attention to application specifications will reveal whether the application-based authentication and authorization mechanisms provide suitable controls to protect against spoofing and browsing. A server application and a client application handshake subroutine may be implemented to ensure that the users are able to employ only the expected software access mechanisms to enter the application. There should be no need to employ other ad-hoc access routines to do this. This is especially true where they may have been developed for application support or administration.

Application-based access controls for penetration are also application subroutines. As is true with the host-based access controls that are designed to prevent system penetration, these are implemented through the use of restricted menus and narrowly defined user groups. Yet, the application users are not defined within the operating system. They are defined within the application itself, so there is a greater risk involved in a successful user exit from the application to the operating system. The application normally will run as an operating system user who has full control over the entire application. Thus, were a user to exit to the operating system, it would be as that privileged user. Application-based access controls for penetration must ensure that any type of exit from the application also disconnects the client from the server. As is the case with host-based access, application-based access controls against the occurrence of eavesdropping are encryption and token-based authentication.

ENCRYPTION MAY BE IMPLEMENTED IN THE HARDWARE COMMUNICATION DEVICE, THE SOFTWARE COMMUNICATIONS PROTOCOL, OR IN THE INFORMATION PROCESSING APPLICATION SOFTWARE.

Controls for DBMS-Based Access

As in the case of application-based access controls for spoofing and browsing, no assumptions should be made concerning the available controls for DBMS access. Many data base management systems do not store passwords in an encrypted format or allow for the encryption of the log-in sequence over the network. Those that do allow some level of encryption should be configured properly to make use of the feature. The more secure DBMS configuration rarely is the default on installation. Some DBMS-based authentication mechanisms will use the password that is supplied by a user operating system account and some will not. If controls against the occurrence

of spoofing and browsing do not exist within the DBMS, they must be custom-written and integrated into the DBMS session connection routines.

DBMS-based access controls against penetration are the restriction of data access to predefined DBMS procedures and the narrow definition of the user groups. Restricting data access to predefined procedures allows an administrator to set permissions by job function. Users are granted access only to those procedures that they should employ to perform their jobs. They are not granted direct access to any data base table. Narrowly defining the size and the composition of the user groups limits the number of the data base procedures to which any user has access. These controls combine to ensure that even if the users are able to penetrate the underlying DBMS they cannot do any more damage to it than they could do through their application screens.

NO ASSUMPTIONS SHOULD BE MADE CONCERNING THE AVAILABLE CONTROLS FOR DBMS ACCESS.

Because DBMS users need not be defined within the operating system and they may be defined only within the DBMS, this DBMS-based access architecture is subject to the same increased penetration vulnerability as that which already has been described for application-based access. The DBMS runs as a privileged system user. Hence, a successful user penetration from the DBMS to the operating system may result in the user gaining full permissions over the entire DBMS. As in the cases of host-based access and application-based access, DBMS-based access controls for eavesdropping are encryption and token-based authentication.

Client/Server Detection Controls

Because the users of a client/server arrangement are dispersed widely, the detection controls for spoofing and browsing start with the controls on the identification of users. Users must protect both their individual log-ins and passwords by not storing them on the client or sharing them among groups with similar job functions. A data base of the user access requests often may be drawn from a customer support application to facilitate frequent automated comparison between the authorized user lists and the server access control lists. Assuming that it is possible to identify the users individually, user log-ins may be monitored for suspicious activity. This activity may include multiple invalid access attempts, simultaneous sessions, and other indications that the person using the log-in is not the authorized user.

Manual processes must ensure that highly suspicious events are investigated and resolved. Systems that have been penetrated are compromised often in such a manner that the perpetrator will find an easy method of access in the future. The perpetrator accomplishes this by altering the system files that pertain to the access mechanisms. A control to detect such client/server penetration is to inventory the system files and to archive them together with algorithmically-derived authentication stamps. The frequent comparison of these archived files will allow for the detection of file tampering.

An additional detection control for penetration is statement level auditing. These types of audit logs should be archived

frequently on a non-production, security-administered system. Doing this will allow an investigator to identify any user log-in that had been compromised to perform the penetration.

Few available detection controls against client/server eavesdropping exist. At most, a skilled communications technician may be able to devise a system that will issue an alert when unknown machines connect to the LAN or WAN. However, even when such an alert system exists eavesdropping from authorized network connections will remain unnoticed.

Client/Server Recovery Controls

The recovery of a client/server application from the effects of a security incident may proceed at either the client or the server level. A robust client/server architecture will be designed so that the client can be completely replaceable. Following a security incident, the memory of a client that has been compromised must be wiped clean and its hard drives reformatted. In addition, a complete copy of the operating system and the software that was being used with it should be installed to replace the compromised platform. No configuration files should be saved, even if they are specific only to an individual user. The clients should be designed so that this process is relatively quick and painless.

At the server level, recovery after a security incident occurs may be more difficult. It may not be possible to decommission the server long enough to reformat and reinstall its contents completely. A statement-level audit of the server content that is copied to a nonproduction, security-administered system will identify the extent of the damage that has occurred. The audit logs must indicate clearly the actions of the individual users. In addition, audit trails must be selectable by user.

The audit logs produce voluminous data, which servers often do not have the capacity to store indefinitely. Restoration of stored media may be required to recover an audit trail or a data base transaction log. Backup mechanisms must allow for targeted restoration of only the affected data to minimize the impact of this reconstruction on already-operating processes. The availability of DBMS table-specific backup or editable transaction logs will speed the recovery process.

AN ADDITIONAL DETECTION CONTROL FOR PENETRATION IS STATEMENT LEVEL AUDITING.

TESTING CLIENT/SERVER SECURITY CONTROLS

Just as the security architecture has the same components regardless of the architecture that is to be secured, controls testing has the same components regardless of the architecture that is to be tested. Despite the flexible nature of the configurations that are to be tested, client/server security audits follow the same outline as those that are carried out with mainframe computers:

- ☐ Standards, policies, and guidelines.
- ☐ Operating system security.
- ☐ Application security.
- ☐ Remote access.
- ☐ Data base management.

- ☐ Program change control.
- ☐ Physical security.
- ☐ Business recovery.

However, the tests that fall under those outline headings are quite different from the control tests on mainframe computers. They will vary considerably between the various client/server applications. To decide what to test for a given client/server application requires research into every component of the application's architecture. The initial steps in performing that research are:

- ☐ Obtaining an architecture diagram and a list of architecture components. (If such detailed schemas are not available, that is the first finding that should be reported under the category of standards, policies, and guidelines.)
- ☐ Reviewing the documentation on the software development tools used to create and maintain the architecture. The development tool standard log-in IDs and the default passwords should be noted.
- ☐ Visiting the development environment and learning the names and nicknames of the machines and the people working there so that they can be recognized if they are encountered again in the production environment.
- ☐ Gathering automated tools for testing the architecture components. A network eavesdropping surveillance tool should be included in this collection.
- ☐ Making a list of the information assets and the tangible assets that are accessible through the targeted application.

THE LIST OF THE ASSETS THAT ARE INVOLVED IS CRITICALLY IMPORTANT IN TESTING THE CLIENT/SERVER ARCHITECTURE'S SECURITY CONTROLS.

The list of the assets that are involved is critically important in testing the client/server architecture's security controls. It is the design and nature of this architecture to provide both accessibility and flexibility to its end users in their pursuit of information services. Hence, demonstrating the absence of a security control, even one that is planned, is not effective unless this demonstration is accompanied by a showing that an asset is at stake. It is possible to use a standard audit plan for each architectural component as a guide in examining the security of the client/server, but the information systems auditor should design all of the tests with the goal of accessing one or more of the assets. It should be remembered that it often is necessary to demonstrate the absence of control in two or more architectural areas to demonstrate that particular assets are at risk.

THE FUNCTION OF CONTROLS IN CLIENT/SERVER

A client/server architecture may be controlled in general by using the same control capabilities that are effective for the respective individual platforms. Hence, maximum use should be made of all of the existing control capabilities at the host, the application, and the DBMS levels. However, the flexibility that is inherent in the client/server architecture also calls for the use of extended controls over access that is provided to it. This may take the form of token authentication, confidentiality in the form of encryption, and auditability in the form of transaction archives. ■

Jennifer Bayuk is a data security practice manager at Price Waterhouse in Morristown NJ. Before joining Price Waterhouse, she was a senior internal auditor at AT&T. Bayuk first joined AT&T as a software engineer at Bell Laboratories.