Implementation of Security in Multi-User Operating Systems

Gabe Emerson

Introduction.

Although most computer systems in use today are of the single-user, personal computer type, there are still many areas where multi-user operating systems are needed and used. The multi-user concept originated during the era of time-sharing mainframe systems, when remote access terminals and more user-friendly interfaces allowed multiple simultaneous connections to and interaction with the systems. During the 1960s and 70s this was the usage model found most often in the computer field, a large central server or mainframe accessed by many dumb terminal workstations, and allocating its resources to each user according to time and precedence. During the 1980s and 90's as more personal computers and smart terminals began to be available to consumers and businesses, the multi-user systems remained a viable model for many large companies, universities, and government agencies. The more powerful descendents of such systems are still in use today among many of these organizations, and in fact the trend among personal computers has begun to belatedly follow that of mainframes. Instead of providing a single user, single-interface terminal, PC operating systems are being tailored to provide separate accounts for different users, more remote access to centralized resources, and other features previously seen only in large business systems. The importance of protecting information and resources on these machines is becoming more important, especially with the spread of home networks and always-on internet connections, which have historically been a popular gateway of attack against larger systems and networks. The need for more public awareness of security needs and methods is growing, and the way operating systems can provide such security and awareness is still evolving. Because Linux/Unix and Microsoft Windows are the most commonly used business and personal operating systems, I will focus on an analysis and comparison of the security features available in each.

Account separation

One of the most important security features of a multi-user operating system is account separation. The ability of each user to have their own private work and storage space within the system is vital for the protection of file integrity and content privacy. It also provides certain conveniences of customized settings and preferences in each account, but these features are minor compared to the security benefits. By allowing each user access to only the files he has created, along with any necessary public files, common tools, and utilities, the system can ensure that users do not interfere with or destroy the work of other users, either purposefully or accidentally. The separation of accounts also assists in monitoring user activity or determining the cause of a problem, since the system can record the actions taken by each users or during the time a particular user is logged in.

Account separation was implemented first in mainframe OS's such as IBM's OS/360 and AT&T's early versions of UNIX. The first personal computer operating system to offer separate user accounts was Linux, the open source OS based on UNIX and including most of UNIX's features. Early PC operating systems such as Apple OS, CPM, Commodore OS, and various DOS clones were primarily designed as single-user systems, with little or no networking features and no account separation (few had any account authorization or login requirements at all). Later offerings by market kingpins Microsoft and Apple continued to be single-user oriented, while Linux and Unix remained the domain of hobbyists and developers. Recently both Microsoft and Apple have begun implementing more and more multi-user features into their products, and today the Windows XP and OS X Operating systems offer nearly as much functionality as Linux in this area.

An important idea in multi-user systems is the concept of permissions. The permission settings of a file or folder determine which users are allowed to view it, who is allowed to change it, and who can create and delete files in certain areas. In Windows, the “Administrator”, who has access to all content on the system, usually controls permissions. Individual users can share or hide their files from each other, or allow changes to be made to files by other users, but the Administrator can read and change anything. In Unix this power is given to the “root” or superuser account, assumed to be the owner or system administrator of the machine. Under Unix/Linux (and recent versions of MacOS based on Unix), permissions are set with a 10-digit permission string assigned to each file. This string consists of four parts, the first digit indicating any special attributes (directory, executable file, etc), the 2nd, 3rd, and 4th digit indicate the permissions given to the file’s owner or creator (owner information is stored elsewhere in the file description). The 5th through the 7th character indicates group permissions, and the 8th through the 10th indicate permissions for every user on the system. Each block of characters can have an r,w, and x, indicating the read, write, and execute permissions.

As noted before, the Administrator or Root accounts are the highest-level accounts on Windows and Unix systems. There may be a need in some systems for different levels of access among other users, for example a business could allow different permissions and privileges to a manager than to a secretary, and a household might want to allow older children more access to the computer than younger children. Both Windows and Unix allow the system owner/administrator to assign various levels of privileges to users, one method is to set up groups of users with similar privileges, so that changes to the group permissions affect only those users in a particular group.

Authentication

Verifying that a user is in fact who they claim to be requires some method of authentication. This is usually done by requiring the user to enter a login name identifying themselves, and a password verifying that they are the owner of that account.

In Linux and Unix, user accounts are stored in an encrypted file located at /etc/passwd. The passwords are encrypted with a randomly generated value resulting in 4,096 possible hashes for each password. However, since the passwd file is readable for all users on the system (a requirement for authentication), it can be copied and compared to values output by a password-cracking program, or “dictionary attack” (such a program takes common words and encodes them with the same encryption algorithm to produce possible password hashes). To protect against such an attack, many Linux distributions implement a system called password shadowing, where the actual password hashes are stored in a location accessible only to the root user, and the passwd file is set up to query this file when it needs to compare an input password with an encrypted password.

Older versions of Microsoft Windows had a very simple username/password authentication, both Windows 98 and 95 did not require any authentication for a default user to log in. Windows 2000 and NT were the first Microsoft products designed explicitly for multiple users, and employed a more secure authentication system. NT-based operating systems (NT, 2000, and XP) store user passwords in a crypted form in the registry, and also in a 56-bit "Data Encryption" hash which provides backwards compatibility with older network authentication methods. Local and remote users can access both of these encrypted files relatively easily, and the encrypted passwords can be extracted and deciphered using a dictionary attack.

It is still possible to configure most operating systems to run in single-user mode (the system assumes one user is always logged in). This can be convenient for a single home PC user, but provides no protection against local tampering and can also disable some of the network protection features.

Logging

Logging is the computer's method of auditing user activity. In early timeshare systems, user activity would be logged so that their use of the CPU and other shared resources could be billed accordingly. Later, logging became useful for recording suspicious or illegal activity on the system, and for troubleshooting problems and errors. Depending on the level of logging set up by the system administrator, anything from the time a user logged in, down to their individual keystrokes, can be recorded. Typical logging simply records the times a user accessed the system, what files they created or modified, what network connections they made, and when they disconnected. More detailed logging can be enabled if necessary, Unix systems support very detailed logging by default, and 3rd-party software is available to supplement this or provide backup records in case there is suspicion of log tampering.

Windows systems usually include only minimal logging by default, including generic network connection logging along with application and system errors. 3-rd party logging packages can increase this functionality to include more detailed information about connection attempts, program and user activity, and detailed application logging. Depending on the degree of monitoring desired, this can include as much or more information than a Linux system's log files support.

Secure Remote connections

A useful aspect of multi-user systems is the ability to connect to the system remotely. In modern usage this is most often done from "smart terminals", or separate, fully functioning computers rather than dumb communications-only terminals. Being able to connect from another computer on the network rather than a dedicated terminal means that the system is open to unauthorized access from other computers, and that communications between terminals and the server can be intercepted or recorded. To improve the security of such communication, many systems limit the types of connections they allow. Typically the telnet protocol is available by default in many Windows and pre-installed Linux systems, but since this connection method uses data sent in plain text it can be intercepted and imitated by unauthorized users. A more secure method is to use ssh, or the secure shell protocol, which encrypts communications traffic between the server and the remote terminal. The operating system can be configured to allow only secure connections, and as a further security measure, to block connection attempts from outside a network of trusted machines. Secure Shell is more commonly used with Unix based systems, and can also be used as a graphical remote interface. Windows systems can also be used as ssh servers with 3-rd party software, or more commonly as vnc (virtual network computing, third-party software), or windows remote access servers, both of which are slightly less secure than ssh.

Common security holes

A backdoor is traditionally a preexisting security hole left by programmers for testing purposes, but the term is also used to mean an access method left by an authorized or unauthorized user. Usually it is a way to bypass the operating system's front-end security measures (such as opening an insecure TCP/IP port), hence the name. Backdoors can be eliminated in the machine's operating system by updating the system as soon as a new backdoor is discovered, or by implementing a software firewall to prevent access to any network ports not specifically known to or utilized by the user.

Network attacks usually take advantage of some published security hole or known backdoor to gain access to a protected part of the system, with the eventual intent of gaining administrative or root access. Often this is done by exploiting a buffer overflow, or passing invalid data to a program and causing it to overflow its assigned memory space and write the attacker's code into another part of the system memory. This method can be used to install and execute a rootkit or similar script, which gains superuser priveledges and allows the attacker to gain control of the system.

Trojans are programs or scripts that authorized users are tricked into using, once executed they do something malicious such as opening a backdoor, running a virus, or collecting user information that an attacker can use to break security measures. Persuading users to run Trojans usually involves some form of social engineering.

Social engineering involves persuading users to give up information about the system, such as login names and passwords. Intruders can trick users into thinking they have a genuine authorization or need for the information, or that they are a repair technician or support worker who is fixing a problem with the user's computer. Social engineering is an often-overlooked area of weakness in security systems, since it is more of a social than a technical aspect. The most effective way to prevent security compromises through this method is to educate users about the dangers of social engineering, and train them not to reveal important security information to unknown persons. Protection against trojans, viruses, spyware, and other voluntary actions that could result in a security compromise is also best brought about through user education.

Local attacks are another often-overlooked part of computer security. By gaining physical access to a system, an attacker can often install malicious software, harvest user accounts and passwords through software or hardware keyloggers, or acquire confidential data from users' accounts. Preventing physical access by intruders is not usually a default feature of operating systems, although most can be configured to minimize this risk. Windows XP does provide default protection of this type, by forcing local users to log back in after a certain time has passed without activity on the system. Linux systems can be configured to do the same through use of a screensaver password or screen lock timeout, but most do not configure this by default. This is one of the few cases where a new Windows XP installation can be more secure than a new generic Linux installation.

However, Windows XP and other version of Windows based on the NT kernel have vulnerability in their registries by which the passwords can be reset or changed. A Windows 200 or XP boot disk can be used to start up the system, log in as the Administrator automatically, and change or reset any passwords. Similarly, several "password reset utilities" based on Linux can be loaded from the floppy drive or CD drive, and used to alter the registry keys containing the password hashes. Linux systems can also be attacked in this way, if an intruder is able to reboot the system from their own disk and access the hard drive, they can copy and possibly compromise the account passwords. Using shadowed passwords can help prevent this type of attack on a Linux system, but no such protection exists natively in Windows. The best defense is to set a BIOS boot password and/or disable the floppy and CD drives and disable booting from external drives.

A recent security compromise at UAF utilized local access to several public lab computers to record passwords and gain control of the systems for illegal purposes. Unfortunately the Department of Computing and Communications was not at liberty to discuss the details of this incident, so the only information I have available is from several anonymous sources. It appears that UAF's security policies do not adequately protect against this type of attack, and the only protection available is analysis of network traffic and investigation of user complaints.

Updating security

To be an effective security tool, any operating system must be kept up to date. Programmers (both benevolent and malicious) are constantly finding vulnerabilities in operating system code, which must be patched to prevent malicious exploitation of the vulnerability once it has become public knowledge. Both Windows and commercial Linux distributions offer automatic updating over the internet, and many websites publish the latest discovered vulnerabilities and information about how to repair or patch them.

Monitoring activities

Logging computer activity has been mentioned, but another method of tracking system use and abuse is by monitoring and logging network activity as well as local system activity. A dedicated computer on the network can act as a logging server, or can hold backups of the logs produced by individual machines. Often this logging is done by a hardware router or firewall device running a proprietary operating system, but it can also be done by software in Linux and Windows environments.

Firewalling involves filtering out unwanted network traffic by blocking certain TCP/IP ports. It attempts to eliminate or at least limit the number or backdoors and attack vectors available to intruders from outside the firewalled network. Intruders on the local network, inside the firewall boundary, can still be a problem. Large networks such as UAF employ a two-tiered approach to network security. In UAF's case, outside network traffic is filtered through a firewall, and undesirable traffic (common attack ports, peer-to-peer file sharing, and other questionable content) is blocked. They also employ a system of heuristic analysis, where the network traffic inside the firewall is analyzed for abnormal or suspicious patterns, and traffic can be tracked back to its source on the network. Some of this analysis takes place at the Operating System level, and is primarily implemented on a UNIX system. However a large part of the tracking and reporting is done by dedicated hardware devices with proprietary operating systems designed specifically for this task. In the case of network logging, Linux/Unix systems usually have many more logging and network utilities installed than do Windows systems, but both can be configured to serve as firewalls or logging systems.

While logging and reviewing logs for interesting patterns is usually a passive monitoring activity, there is another method which takes a more active approach to observing illicit activity over a network. A “honeypot” can be used to attract crackers and observe their activities to learn more about typical attacks and how to find and deal with security compromises. This is usually done by setting up a system with a very basic OS installation, with known vulnerabilities and security holes, and then observing any attacks carried out against it. On a large network such as UAF’s there are usually at least a few people running port scanning and packet sniffing programs in an effort to find such vulnerable machines and compromise them. The legality of using honeypots has been questioned in a few cases, since legally it can be seen as a form of entrapment. If used merely for observation and not for prosecution of attackers, then honeypotting should be a perfectly legal way to learn about security holes in the chosen operating system.

Conclusion.

Securing a computer is a time consuming process no matter what operating system is chosen. While the two most popular multi-user operating systems are quite different in their implementation of security and the steps necessary to adequately protect them, they can each be brought to an equivalent level of security through careful observance of a few standard security policies and the addition or configuration of some necessary software tools. In terms of native security, Unix based operating systems are much more secure after a typical installation, but the level of knowledge and skill needed to fully implement a trustworthy environment is higher than that needed for the more user-friendly Windows-based operating systems. Windows may be easier to configure, but the tools needed to bring it up to the same level of protection offered by Linux and Unix are not typically included in a new installation and must be purchased or downloaded from third-party vendors. This can take more time and effort than a Linux system, and require more attention to detail to avoid the many security holes in a basic Windows installation. Because of the relatively low security knowledge of typical Windows users, keeping a system secure and up to date can be more difficult and require more user education than would a work force of experienced Linux users. Each Operating system has its benefits and drawbacks, so the choice of which to implement is largely based upon the needs, desires, and resources of the particular user or company.

References:

Bace, Rebecca Gurley. Intrusion Detection. Indianapolis: Macmillan, 2000.

"History of Operating Systems." Wikipedia. March 2004
http://en.wikipedia.org/wiki/History_of_operating_systems

Mandia, Kevin, and Chris Prosise. Incident Response: Investigating Computer Crime.
New York: McGraw Hill, 2001.

Medbury, Todd. Personal Interview. 26 Apr. 2004.

Nordahl, P. "Offline NT Password and Registry Editor". 2004.
http://home.eunet.no/~pnordahl/ntpasswd/

Redding, Loren E, ed. Linux Complete. 2nd ed. Alameda: Sybex Inc, 2002.
Northcutt, Stephen, and Judy Novak. Network Intrusion Detection. 3rd ed. New York: New Riders Publishing, 2003.

Tanenbaum, Andrew S. Modern Operating Systems. 2nd ed. New Jersey: Prentice-Hall, 2001.

"The Hack FAQ: Answers to Frequently Asked Questions About Hacking" 2600 Magazine. 2001. http://www.hackfaq.org/