You can use the *Event Viewer* utility to track information about your computer's hardware and software, as well as to monitor security events. Every Windows XP computer will show three types of log files (depending on your configuration you may also have other log files):

System log

Tracks events related to the Windows XP operating system. This log is useful in troubleshooting Windows XP problems.

Security log

Tracks events related to Windows XP auditing. By default auditing is not enabled. If you enable security auditing you can select what to track and whether you will track security success and/or failure events.

Application log

Tracks events related to applications that are running on your computer. For example, you might see that your e-mail program recorded an error. These errors are useful in troubleshooting application problems or for developers to fix application problems.

INTERRUPT

Interrupts are central to operating systems as they provide an efficient way for the operating system to interact and react to its environment. The alternative is to have the operating system "watch" the various sources of input for events (polling) that require action -- not a good use of CPU resources. Interrupt-based programming is directly supported by most CPUs. Interrupts provide a computer with a way of automatically running specific code in response to events. Even very basic computers support hardware interrupts, and allow the programmer to specify code which may be run when that event takes place.

When an interrupt is received the computer's hardware automatically suspends whatever program is currently running, saves its status, and runs computer code previously associated with the interrupt. This is analogous to placing a bookmark in a book when someone is interrupted by a phone call and then taking the call. In Windows operating systems interrupts are handled by the operating system's kernel. Interrupts may come from either the computer's hardware or from the running program.

kernel is the core process of a preemptive operating system, consisting of a multitasking scheduler and the basic security services. Depending on the operating

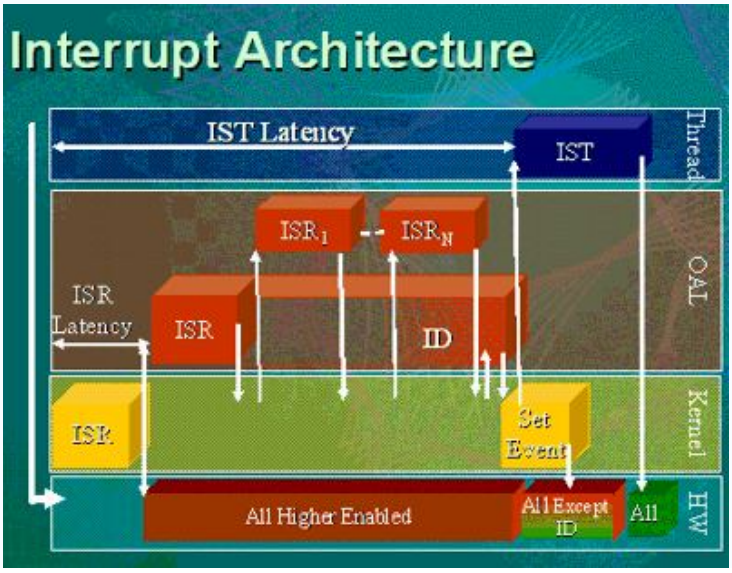
system, other services such as virtual memory drivers may be built into the kernel. The kernel is responsible for managing the scheduling of threads and processes.

When a hardware device triggers an interrupt the operating system's kernel decides how to deal with this event, generally by running some processing code. How much code gets run depends on the priority of the interrupt (for example: a person usually responds to a smoke detector alarm before answering the phone). The processing of hardware interrupts is a task that is usually delegated to software called device drivers, which may be either part of the operating system's kernel, part of another program, or both. Device drivers may then relay information to a running program by various means.

A program may also trigger an interrupt to the Windows operating system. If a program wishes to access hardware for example, it may interrupt the operating system's kernel, which causes control to be passed back to the kernel. The kernel will then process the request. If a program wishes additional resources (or wishes to shed resources) such as memory, it will trigger an interrupt to get the kernel's attention.

Interrupt Architecture

The first step in exploring the interrupt architecture of Microsoft® Windows® is defining an overall model of the hardware, kernel, OAL and thread interactions during an interrupt. The following diagram is an overall picture of these different levels of responsibility and the transitions that cause changes of state.



The diagram represents the major transitions during an interrupt with time increasing to the right of the diagram. The bottom most layer of the diagram is the hardware and the state of the interrupt controller. The next layer is the kernel interactions during interrupt servicing. The OAL describes the board support package (BSP) responsibilities. The top most layer represents the application or driver thread interactions needed to service an interrupt. The diagram represents the interactions during a single interrupt; representing the new ability of Windows to have shared interrupts. The activity starts with an interrupt represented by the line at the left most section of the chart. An exception is generated causing the kernel ISR vector to be loaded onto the processor. The kernel ISR interacts with the hardware disabling all equal and lower priority interrupts on all processors except for the ARM and Strong ARM architectures. The kernel then vectors to the OAL ISR that has been registered for that particular interrupt. The OAL ISR then can either directly handle the interrupt or can use NKCallIntChain to walk a list of installed ISRs. The main ISR or any of the installed ISRs then performs any work and returns the mapped interrupt called SYSINTR for that device. If the ISR determines that its associated device is not causing the interrupt the ISR returns SYSINTR_CHAIN, which causes NKCallIntChain( ) to walk the ISR list to the next interrupt in the chain. The ISRs are called in the order that they were installed creating a priority on the calling list.

Microsoft has advanced the Windows interrupt architecture. The ability of the OS to deal with shared interrupts greatly extends the ability of Windows to support many interrupt architectures. Knowledge of this interrupt architecture greatly speeds up the investigation times into driver and latency issues. A Windows model of the operating system interaction is the key to this understanding. Shared interrupts have greatly increased the openness of Windows supporting the platform provider and application developer scenarios that are pervasive from company to company or within companies. Understanding latency sources will help in diagnosis of driver and real-time issues. The interrupt structure in Windows is well defined and understandable.

SECURITY

Security has been a hot topic with Windows for many years, and even Microsoft itself has been the victim of security breaches.[citation needed] Consumer versions of Windows were originally designed for ease-of-use on a single-user PC without a network connection, and did not have security features built in from the outset.

Windows NT and its successors are designed for security (including on a network) and multi-user PCs, but were not initially designed with Internet security in mind as much since, when it was first developed in the early 1990s, Internet use was less prevalent.
These design issues combined with flawed code (such as buffer overflows) and the popularity of Windows means that it is a frequent target of computer worm and virus writers.
Microsoft releases security patches through its Windows Update service approximately once a month (usually the second Tuesday of the month), although critical updates are made available at shorter intervals when necessary. In Windows 2000 (SP3 and later), Windows XP and Windows Server 2003, updates can be automatically downloaded and installed if the user selects to do so.

Use these steps to develop a strong password:

- Think of a sentence that you can remember. This will be the basis of your strong password or pass phrase. Use a memorable sentence, such as "My son Aiden is three years old."
- Check if the computer or online system supports the pass phrase directly. If you can use a pass phrase (with spaces between characters) on your computer or online system, do so.
- If the computer or online system does not support pass phrases, convert it to a password. Take the first letter of each word of the sentence that you've created to create a new, nonsensical word. Using the example above, you'd get: "msaityo".

- Add complexity by mixing uppercase and lowercase letters and numbers. It is valuable to use some letter swapping or misspellings as well. For instance, in the pass phrase above, consider misspelling Aiden's name, or substituting the word "three" for the number 3. There are many possible substitutions, and the longer the sentence, the more complex your password can be. Your pass phrase might become "My SoN Ayd3N is 3 yeeRs old." If the computer or online system will not support a pass phrase, use the same technique on the shorter password. This might yield a password like "MsAy3yo".
- Finally, substitute some special characters. You can use symbols that look like letters, combine words (remove spaces) and other ways to make the

password more complex. Using these tricks, we create a pass phrase of "MySoN 8N i$ 3 yeeR$ old" or a password (using the first letter of each word) "M$8ni3y0".

- Test your new password with Password Checker Password Checker is a non-recording feature on this Web site that helps determine your password's strength as you type.

✓ **4 steps to protect your computer**

Step 1. Keep your firewall turned on

What is a firewall?

A firewall helps protect your computer from hackers who might try to delete information, crash your computer, or even steal your passwords or credit card numbers. Make sure your firewall is always turned on.

Step 2. Keep your operating system up-to-date

What are operating system updates?

High priority updates are critical to the security and reliability of your computer. They offer the latest protection against malicious online activities. Microsoft provides new updates, as necessary, on the second Tuesday of the month.

Step 3. Use updated antivirus software

What is antivirus software?

Viruses and spyware are two kinds of usually malicious software that you need to protect your computer against. You need antivirus technology to help prevent viruses, and you need to keep it regularly updated.

Step 4. Use updated antispyware technology

What is antispyware software?

Viruses and spyware are two kinds of usually malicious software that you need to protect your computer against. You need antispyware technology to help prevent spyware, and you need to keep it regularly updated.

Some threats to the security of Windows are;

I. ***Worms***
II. ***Spyware***
III. ***Viruses***

- ✓ **Security in Microsoft Windows**

While the Windows 9x series offered the option of having profiles for multiple users, they had no concept of access privileges, and did not allow concurrent access; and so were not true multi-user operating systems. In addition, they implemented only partial memory protection. They were accordingly widely criticised for lack of security.

The Windows NT series of operating systems, by contrast, are true multi-user, and implement absolute memory protection. However, a lot of the advantages of being a true multi-user operating system were nullified by the fact that, prior to Windows Vista, the first user account created during the setup process was an administrator account, which was also the default for new accounts. Though Windows XP did have limited accounts, the majority of home users did not change to an account type with fewer rights – partially due to the number of programs which unnecessarily required administrator rights – and so most home users ran as administrator all the time.

Windows Vista changes this by introducing a privilege elevation system called User Account Control. When logging in as a standard user, a logon session is created and a token containing only the most basic privileges is assigned. In this way, the new logon session is incapable of making changes that would affect the entire system.

When logging in as a user in the Administrators group, two separate tokens are assigned. The first token contains all privileges typically awarded to an administrator, and the second is a restricted token similar to what a standard user would receive. User applications, including the Windows Shell, are then started with the restricted token, resulting in a reduced privilege environment even under an Administrator account. When an application requests higher privileges or "Run as administrator" is clicked, UAC will prompt for confirmation and, if consent is given (including administrator credentials if the account requesting the elevation is not a member of the administrators group), start the process using the unrestricted token.

Windows Defender

On January 6, 2005, Microsoft released a beta version of Microsoft AntiSpyware, based upon the previously released Giant AntiSpyware. On February 14, 2006, Microsoft AntiSpyware became Windows Defender with the release of beta 2. Windows Defender is a freeware program designed to protect against spyware and other unwanted software. Windows XP and Windows Server 2003 users who have genuine copies of Microsoft Windows can freely download the program from Microsoft's web site, and Windows Defender ships as part of Windows Vista.

File Permissions

At windows version from window NT3 have been based on a file system permission system referred to as AGLP (Account, Global, Local, and Permission). In essence, where file permissions are applied to the file and folder in the form of a local group, which then has other global group.

CHAPTER NINE

INTRODUCTION

The fact that an operating system is computer software makes it prone to error just as any human creation. Programmers make mistakes, and inefficient code is often implemented into programs even after testing. Some developers perform more thorough testing and generally produce more efficient software. Therefore, some operating systems are more error prone while others are more secure.

1.1Security in Computer

The branch of computer technology known as information security as applied to computers and networks is the computer security. The objective of computer security includes protection of information and property from theft, corruption, or natural disaster, while allowing the information and property to remain accessible and productive to its intended users. The term computer system security means the collective processes and mechanisms by which sensitive and valuable information and services are protected from publication, tampering or collapse by unauthorized activities or untrustworthy individuals and unplanned events respectively.

The technologies of computer security are based on logic. As security is not necessarily the primary goal of most computer applications, designing a program with security in mind often imposes restrictions on that program's behavior.
Security in this regard could be referred to as the protection mechanism used to safeguard information in the computer. Security has many facets and the three of the more important ones are

1. The threats
2. The nature of intruders
3. Accidental data loss

1. **Threats**: From security perspective, computer systems have three general goals, with corresponding threats.

(i) Data Confidentiality: Concerned with having secret data remain secret meaning if the owner of some data has decided that these data are available to certain people and no others, the system should guarantee that release of the data to unauthorized people does not occur.

(ii) Data Integrity: Means that unauthorized users should not be able to modify any data without the owner's permission. If a system cannot guarantee that data deposited in it remain unchanged until the owner decides to change them, it is not worth much as an information system.

(iii) System Availability: Means that nobody can disturb the system to make it usable. If a computer is an internet server, sending a flood of requests to it may cripple it by eating up all of its CPU time just examining and discarding incoming requests. Another aspect of security problem in operating system is Privacy that is protecting individuals from misuse of information about them.

(2) **The nature of intruders:** In the security literature, people who are nosing around places where they have no business being are called ***intruders*** or sometimes called ***adversaries.*** Intruders act in two different ways

1 ***Passive Intruders****:* just want to read files they are not authorised to read.

2 ***Active Intruders***: are more malicious; they want to make unauthorised changes to data.

When designing a system to be secure against intruders, it is important to keep in mind the kind of intruder one is trying to protect against. Some categories are

(a) Casual prying by nontechnical users.

(b) Snooping by insiders.

(c) Determined attempts to make money.

(d) Commercial or military espionage.

Another category of security pest that has manifested itself in recent times is the ***Virus***. The term virus is a piece of code that replicates itself and usually does some damage.

(3) **Accidental data loss:** In addition to threats caused by malicious intruders, valuable data can be lost by accident. Some of the accidental data losses are

(1) ***Acts of God***: fires, flood, earthquakes, wars, etc.

(2) ***Hardware or software errors***: CPU malfunction, unreadable disks or tapes, telecommunication errors, program bugs.

(3) ***Human errors:*** incorrect data entry, wrong tape or disk mounted, wrong program run, lost disk or tape, or some other mistake.Most of these can be dealt with by maintaining adequate backups, preferably far away from the original data.

1.2 Operating system design, security and complexity

An operating system is a software component that acts as the core of a computer system. It performs various functions and is essentially the interface that connects your computer and its supported components. Various operating systems are in use today to satisfy the ever changing customer demands. Nevertheless the most widespread operating systems are: **Microsoft Windows, Linux/Unix and Macintosh**. **Windows** are mostly used as personal computers, **Linux/Unix** are mainly open source while **Apple Macs** are often used for graphic designs or other specialist applications. Irrespective of their application or use, all operating systems up to date have been subject to security compromises or likewise failures. It is a fact that the majority of hacking tools, viruses, worms or Trojan horses are written for Windows, but this is merely due to the fact that Windows occupy almost 90% of the global market.

The security of the operating system is therefore a necessity for the overall system security. Today most commercially developed operating systems provide security through authentication of the users, maintenance of access control mechanisms, separation of kernel and user spaces and providing trusted applications to modify or manage system resources. However the above security features are inadequate to protect the operating system from attacks in today's environment.

SECURITY ISSUES IN THE LINUX OPERATING SYSTEMS

While companies like Microsoft and Apple own the commercial software market, a variety of non profit organizations or intellectual individuals contribute constantly to the proliferation of the ***open source software***. In terms of operating systems, Linux is the example par excellence of all free ***open source systems***, Introduced by **Linux Torvalds**, at the time a student. Linux was the first fully functional operating system that was offered for free under the ***open source*** agreement to the public. Following this event,

with the participation of thousand of admirers across the globe, a myriad of ***open source*** Linux based operation systems flooded the cyber world.

There are different reasons that motivate thousands of people around the globe to participate in open source projects and release software to the public. Intellectual gratification, pleasure of creativity or of solving complex and challenging tasks, are of some the driving forces in this domain. Whatever the reasons, the benefits of using open source are manifold. Open-source software powers many of the web sites on the Internet, corporate computer, servers used for research and development, it can be found in digital video recorders , telephones, personal digital assistants (PDAs) watches, networking hardware, MP3 players and automobiles

Nevertheless, open source software does not come without issues or disadvantages. Even though the codes are available to thousands of eyes for scrutiny, there is no guarantee for security or optimal performance. Although that is ok for simple home applications it is not the case for enterprise, commercial or critical applications. Also because of lack of standardization and complex licensing issues, open source software is prone to misuse or abuse. Standardization is hard to achieve, because open source creators are completely free in their choice of design, implementation or adherence to existing standards. Usually standardization is enforced by market forces and industry regulators; however in the case of open source software both these factors do not exercise enough pressure to drive the process. Version proliferation is another major open source issue. As a matter of fact, this matter does not concern only open source software but commercial software as well. Nevertheless, the effect on the open source software is more evident. Developing many versions of a program in a short period not only confuses users but also requires a steep learning curve. This is true in particular in the case of constant changes of graphical user interfaces and navigation concepts from one version to the next one. At the same time, constant introductions of new versions of a software package do not affect in positive way its reputation since the user might believe this is a sign of instability.

Another issue that affects ICT today is that of the implementation of open source software. Because, open source is developed by the co-operation of different individuals, it is hard to establish a proper working relationship with the person in charge (if there is any). Furthermore, technical support, documentation issues, no access to advice, are some of the problems that a

company or individual that uses open source software might face. Hardware and software compatibilities influence also the processes ever since most of hardware manufacturers do not expose their trade secrets, therefore not allowing access to their codes to open source developers. Open source developers have no other choice but to design and release their own code (hardware drivers) therefore contributing to the complexity and lack of standardization.

Future prognosis on the Linux operating system

The open source phenomenon is definitely influencing in a positive way ICT and probably the trend will not change in the future. Open source projects are available to "millions of eyes" for scrutiny, improvement or testing. Nevertheless, it is likely that in the future will continue to experience the same issues mentioned above with some improvements in the area of standardization. Some open source will definitely transform in commercial software provided that they have matured enough and captured a significant market share. Red Hat Linux for example, is a typical example of how open source software becomes commercial under the right circumstances.

SECURITY ISSUES IN THE WINDOWS OPERATING SYSTEM

Security is the main problem that Windows operating systems are facing since their introduction. Lack of vision from its developers regarding security is probably the main reason behind this issue. The first windows were designed to be simple and productive but not very secure. Although new operating systems versions were introduced within the last 5-10 years, the same issues with security persisted. In my opinion, because of market pressure and product development circles, it was almost impossible for Microsoft to totally change their operating system approach. Instead they continued to build on top of each previous model. Unfortunately, their operating systems are still vulnerable affecting significantly ICT applications worldwide.

To understand the impact of operating system vulnerabilities on ICT suffice to look at the case of "SQL slammer', a worm released on the web in 2003 (Forte, 2003). Slammer, also known as "SQL hell", is a worm that affected Microsoft Windows operating systems in January 2003 affecting within ten minutes 75 thousands machines worldwide. Slammer exploited vulnerability

in SQL server and desktop engine slowing down communications and affecting businesses financially worldwide. Following this incident several modified slammer versions were released online. This is not an isolated episode that demonstrates the lack of security vision and poor operating system design. Viruses and worms like Melissa, code red, sasser, nimda, donut, spida or slapper have also impacted information communication telecommunications globally. In 2007 Computer Economics, a well known research company conducted a research on the impact of the malware globally estimating a $ 13 billion in financial losses in 2006 only. It is quite obvious how ICT and global communications are affected by malware which attributes its success to operating system vulnerabilities.

Microsoft has been able to take care of malware attacks by releasing patches or services packs. Although, it looks like this is the right approach to this problem it does eradicate the problem and provide temporarily relief from threats. We need a holistic approach to deal with the root of the problem not with it consequences. As a matter of fact operating system patches manage to avoid the threat temporarily, since what they usually do is a mere change of names or locations of important operating system files used by malware. In other occasions Microsoft has even discontinued shipping certain programs with its operating systems as a security measure against malware, therefore not dealing with the main problems: design and security.

In 2007, Microsoft released officially to the public Windows Vista and in October 2009 Windows seven. Although, the graphical user interfaces look impressive, both operating systems are still vulnerable to malware or system hacking. A very simple example is the ‘the sticky key backdoor’, one of Vista’s vulnerabilities. Vinoo Thomas, a McAfee researcher, in 2007 released a blog online informing the public about the Sticky Keys vulnerability. Vista apparently does allow the modification of sethc.exe file (located at: C:/windows/system32/sethc.exe) and no integrity checks are performed before execution. **Authentication** can simply be bypassed by replacing this file with cmd.exe using a live CD like Backtrack or direct logging and entering windows explorer (Vinoo, 2007).

Moreover, Vista activation mechanism has been broken almost one year after its official release. The same security scenario applies to Windows seven. This operating system can be bypassed in the same way as Windows Vista (using the installation disk and entering recovery mode). Furthermore

online news of a zero day attack is spreading around. Certainly, this poor security performance of Microsoft Windows, the most used operating system worldwide, does not sound promising for ICT and its future.

Future prognosis on the windows operating system

Operating system design is a factor that will influence ICT in the times to come. The main reason is security. If we take in consideration the fact that digital globalization is facilitating the distribution of malware and the number of internet users is rising exponentially, we should expect more sophisticated attacks on the windows operating systems and ICT. Under these circumstances, we should review the windows operating system design strategy, focusing on security and build reliable operating system.

CHAPTER NINE:
SECURITY ISSUES IN THE UNIX OPERATING SYSTEMS

There are a number of elements that have lead to the popularity of the UNIX operating system in the world today. The most notable factors are its portability among hardware platforms and the interactive programming environment that it offers to users. In fact, these elements have had much to do with the successful evolution of the UNIX system in the commercial market place. As the UNIX system expands further into industry and government, the need to handle UNIX system security will no doubt become imperative. For example, the US government is committing several million dollars a year for the UNIX system and its supported hardware. The security requirements for the government are tremendous, and one can only guess at the future needs of security in industry. In this paper, we will cover some of the more fundamental security risks in the UNIX system. Discussed are common causes of UNIX system compromise in such areas as file protection, password security, networking and hacker violations. In our conclusion, we will comment upon ongoing effects in UNIX system security and their direct influence on the portability of the UNIX operating system.

In the UNIX operating system environment, files and directories are organized in a tree structure with specific access modes. The setting of these modes through permission bits (as octal digits), is the basis of UNIX system security. Permission bits determine how users can access files and the type of access they are allowed. There are three user access modes for all UNIX system files and directories: the owner, the group, and others. Access to read, write and execute within each of the user types is also controlled by permission bits. Flexibility in file security is convenient, but it has been criticized as an area of System security compromise.

FILE AND DIRECTORY SECURITY

In the UNIX operating system environment, files and directories are organized in a tree structure with specific access modes. The setting of these modes, through permission bits (as octal digits), is the basis of UNIX system security. Permission bits determine how users can access files and the type of access they are allowed. There are three user access modes for all UNIX system files and directories: the owner, the group, and others. Access to read, write and execute within each of the user types is also controlled by permission bits (Figure 1). Flexibility in file

security is convenient, but it has been criticized as an area of system security compromise.

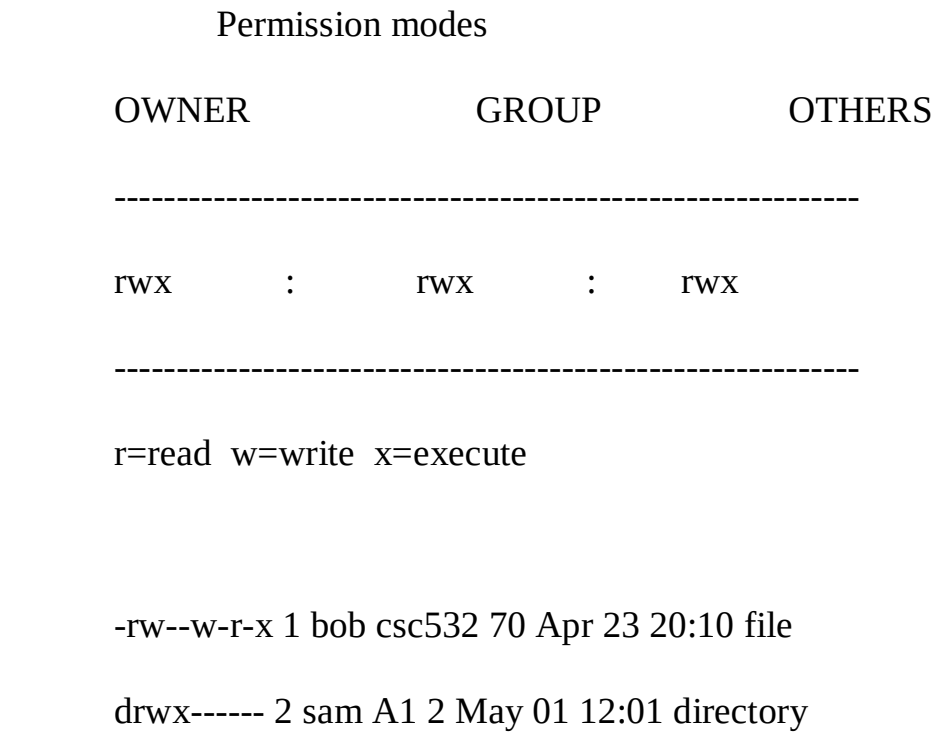


FIGURE 1. File and directory modes: File shows Bob as the owner, with read and writes permission. Group has write permission, while others has read and execute permission. The directory gives a secure directory not readable, writeable, or executable by Group and Others.

Since the file protection mechanism is so important in the UNIX operating system, it stands to reason that the proper setting of permission bits is required for overall security. Aside from user ignorance, the most common area of file compromise has to do with the default setting of permission bits at file creation. In some systems the default is octal 644, meaning that only the file owner can write and read to a file, while all others can only read it. (3) In many "open" environments this may be acceptable. However, in cases where sensitive data is present, the access for reading by others should be turned off. The file utility umask does in fact satisfy this requirement. A suggested setting, umask 027, would enable all permission for the file owner, disable write permission to the group, and disable permissions for all

others (octal 750). By inserting this umask command in a user .profile or .login file, the default will be overwritten by the new settings at file creation. The CHMOD utility can be used to modify permission settings on files and directories. Issuing the following command,

```
chmod u+rwd,g+rw,g-w,u-rwx file
```

will provide the file with the same protection as the umask above (octal 750). Permission bits can be relaxed with chmod at a later time, but at least initially, the file structure can be made secure using a restrictive umask. By responsible application of such utilities as umask and chmod, users can enhance file system security. The Unix system, however, restricts the security defined by the user to only owner, group and others. Thus, the owner of the file cannot designate file access to specific users. As Kowack and Healy have pointed out, "The granularity of control that (file security) mechanisms is often insufficient in practice (...) it is not possible to grant one user write protection to a directory while granting another read permission to the same directory. (4) A useful file security file security extension to the Unix system might be Multics style access control lists. With access mode vulnerabilities in mind, users should pay close attention to files and directories under their control, and correct permissions whenever possible. Even with the design limitations in mode granularity, following a safe approach will ensure a more secure Unix system file structure.

DIRECTORIES

Directory protection is commonly overlooked component of file security in the Unix system. Many system administrators and users are unaware of the fact, that "publicly writable directories provide the most opportunities for compromising the Unix system security" (6). Administrators tend to make these "open" for users to move around and access public files and utilities. This can be disastrous, since files and other subdirectories within writable directories can be moved out and replaced with different versions, even if contained files are unreadable or unwritable to others. When this happens, an

unscrupulous user or a "password breaker" may supplant a Trojan horse of a commonly used system utility' *For example:*

Imagine that the /bin directory is publicly writable. The perpetrator could first remove the old version (with rm utility) and then include his own fake su to read the password of users who execute this utility.

Although writable directories can destroy system integrity, readable ones can be just as damaging. Sometimes files and directories are configured to permit read access by other. This subtle convenience can lead to unauthorized disclosure of sensitive data: a serious matter when valuable information is lost to a business competitor.
As a general rule, therefore, read and write access should be removed from all but system administrative directories. Execute permission will allow access to needed files; however, users might explicitly name the file they wish to use. This adds some protection to unreadable and unwritable directories. So, programs like lp file.x in an unreadable directory /ddr will print the contents of file.x, while ls/ddr would not list the contents of that directory.

USER AUTHENTICATION

Another area is the user authentication. In the UNIX system, authentication is accomplished by personal passwords. Though passwords offer an additional level of security beyond physical constraints, they lend themselves to the greatest area of computer system compromise. Lack of user awareness and responsibility contributes largely to this form of computer insecurity. This is true
of many computer facilities where password identification, authentication and authorization are required for the access of resources, and the Unix operating system is no exception. Password information in many time-sharing systems are kept in restricted files that are not ordinarily readable by users. The UNIX system differs in this respect, since it allows all users to have read access to the /etc/passwd file where encrypted passwords and other user information are stored.

DATA ENCRYPTION

Although the Unix system implements a one-way encryption method, and in most systems a modified version of the data encryption standard (DES), password breaking methods are known. Among these methods, brute-force attacks are generally the least effective, yet techniques involving the use of heuristics (good guesses and knowledge about passwords) tend to be successful. For example, the /etc/passwd file contains such useful information as the login name and comments fields. Login names are especially rewarding to the "password breaker" since many users will use login variants for passwords (backward spelling, the appending of a single digit etc.). The comment field often contains items such as surname, given name, address, telephone number, project name and so on. To quote Morris and Grampp in their landmark paper on Unix system security: The authors made a survey of several dozen local machines, using as trial passwords a collection of the 20 most common female first names, each followed by a single digit. The total number of passwords tried was therefore 200. At least one of these 200 passwords turned out to be a valid password on every machine surveyed. If an intruder knows something about the people using a machine, a whole new set of candidates is available. Family and friend's names, auto registration numbers, hobbies, and pets are particularly productive categories to try interactively in the unlikely event that a purely mechanical scan of the password file turns out to be disappointing.

Thus, given a persistent system violator, there is strong evidence, that he will find some information about users in the /etc/passwd file. With this in mind, it is obvious that a password file should be unreadable to everyone except those in charge of system administration.

SECURITY ISSUES IN THE MACINTOSH OPERATING SYSTEM

It's been called one of the safest operating systems of all times, but the Mac's OS X Tiger may not be as safe as it seems. Mac's OS X Tiger has become a favorite among Mac users for its bells and whistles and its UNIX based architecture. From a power user to newbie, Tiger provides both comfort and security for all OS X users. Some of the flaws found in its security is as follows:

- *FAILING TO USE ITS SOFTWARE UPDATE*

Regularly updating Tiger's software is one of the easiest ways to keep your computer protected from the latest exploits and malicious Internet content. In January of this year, a couple of computer guru's published "The Month of Apple Bugs"(MoAB) -- a website dedicated to pointing out 31 of OS X's vulnerabilities and security flaws. After reviewing the website, Apple acted promptly and has since released several updates addressing the critical bugs. With software updates turned off, there's a good chance the computer could fall victim to one of MoAB's exploits.

- *MINDLESSLY SURFING WITH SAFARI*

Although much safer than Microsoft's Internet Explorer, Tiger's default web browser Safari is not immune to security flaws. To obtain the safest Internet browsing experience, a few of Safari's features should be modified: Make sure all "AutoFill" options are disabled, and always use "Private Browsing" on each of the computer's accounts. Although surfing without "Private Browsing" enabled could save you some time, in the long run you're simply opening yourself up to greater security risks.

To be ultra conservative with web browsing, one can disallow all cookies and remove all existing cookies via the "Show Cookies" dialog. (Keeping in mind that some websites require cookies for complete functionality). By not accepting cookies, one may be limiting web browsing experience, so one is torn in between securing his/her system or enjoying the web experience.

- *INCORRECTLY CONFIGURING SECURITY PREFERENCES*

Tiger's security panel features a handful of security preferences which permit users to select varying levels of security based upon their particular usage requirements. Obviously, however, when configuring your security preference it's important to understand what each option does, and the benefits of a particular setting: One of the most important and often overlooked preferences is whether to permit automatic login. Requiring a password to wake the computer is imperative in preventing unauthorized access to unattended computers. "Disabling automatic login is necessary for any level of security. If you enable automatic login, an intruder can automatically log in without having to authenticate. Even if you automatically log in with a very restricted user account, this makes it much easier to perform malicious actions on the computer."

In addition to requiring a user to login after the computer has been asleep, it is also important to require an additional login wherever an important system

wide preference is being changed. In order to prevent faulty administration, either from a malicious user or just from an unwitting friend who accidentally makes a system change, it is important to require an extra step of authentication when altering system preferences. After all, we can all make mistakes when toggling options, but by requiring an extra authentication step whenever a system preference is changed, you can make sure that many of these types of errors never occur.

- *LEAVING UNUSED HARDWARE DEVICES ENABLED*

Most of us are no longer worried by our Internet connections, but instead connect to the Internet through multiple styles of broadband connections. For example, instead of being tied down to Ethernet cables at home, users are taking advantage of wireless connections using their laptops from anywhere, and connecting their Bluetooth devices to their computer for extra support. While having different types of broadband connections is great for the user, it's awful for the security of the computer. To protect your computer, you should make sure to "disable any unused hardware devices listed in Network preferences. Enabled, unused devices (such as AirPort and Bluetooth) are a security risk." While we're not suggesting that you do away with these great feature altogether, we are saying that when they're not in use, you should turn them off.

- *TROJAN HORSE ALERT*

Recently, a new variant of the Hell Raiser Trojan Horse, which was identified as OSX/HellRTS.D, has been discovered. Experts have analyzed this new variant, and it is detected in the latest MacScan spyware definitions update as HellRaiser Trojan Horse 4.2. MacScan has detected previous variants of this trojan horse since 2005. HellRaiser is a trojan horse that allows complete control of a computer by a remote attacker, giving the attacker the ability to transfer files to and from the infected computer, pop up chat messages on the infected system, display pictures, speak messages, and even remotely restart or shut down the infected machine. The attacker can search through the files on the infected computer, choosing exactly what they want to steal, view the contents of the clipboard, or even watch the user's actions on the infected computer.

In order to become infected, a user must run the server component of the Trojan horse, which can be disguised as an innocent file. The attacker then

uses the client component of the Trojan horse to take control of the infected system.

SECURITY ISSUES IN SOLARIS OPERATING SYSTEM

Solaris or Oracle Solaris as it is now known is a UNIX-based operating system introduced by Sun Microsystems in 1992 as the successor to SunOS. The prominent flaw in Solaris operating system is the multiple security vulnerabilities in PostgreSQL Shipped with Solaris 10 which allows the Elevation of Privileges or Denial of Service (DoS)

Multiple Security vulnerabilities affecting the PostgreSQL software shipped with Solaris 10 may allow a local or remote user who has access to the PostgreSQL server to cause a Denial of Service (DoS) to the PostgreSQL instance or the server it runs on (due to excessive resource consumption), or to gain elevated privileges on the server.

Regular Expression Denial-of-Service

(CVE-2007-4772, CVE-2007-6067, CVE-2007-4769):

Three separate issues in the regular expression libraries used by PostgreSQL allow malicious user to initiate a denial-of-service by passing certain regular expressions in SQL queries.

First, users could create infinite loops using some specific regular expressions.

Second, certain complex regular expressions could consume some excessive amounts of memory.

Third, out-of-range backref numbers could be used to crash the backend.

All of these issues have been patched.

DBLink privilege Escalation (CVE-2007-6601):

DBLink functions combined with local trust or ident authentication could be used by malicious user to gain superuser privileges.

This issue has been fixed and does not affect users who have not installed DBLink (an optional module), or who are using password authentication for local access. This same problem was addressed in the previous release cycle.

These issues can occur in the following releases

SPARC (Scalable Processor Architecture) Platform

Solaris 10 PostgreSQL 8.1

Without patch 123590-08

Solaris 10 PostgreSQL 8.2

Without patch 136998-02

X86 Platform

Solaris 10 PostgreSQL 8.1
Without patch 123591-08
Solaris 10 PostgreSQL 8.2
Without patch 136999-02

Solaris 8 and 9 do not ship with PostGreSQL and are not impacted by this issue.
A user exploiting this vulnerability must have an account on the PostgreSQL server.
This issue affects PostGreSQL versions 7.4x prior to 7.4.19, 8.0.x prior to 8.0.15, 8.1.x prior to 8.1.11 and 8.2.x prior to 8.2.6.

CONCLUSION

Information and Communication Technologies (ICT) will provide benefits to our society for years to come. The proliferation of these technologies or their decline will be affected amongst all by security issues on the area of operating system design and security, open source issues, and design complexity. Therefore, designing better operating systems, improving on their security, are some of the challenges for the future. If we take in consideration the fact that digital globalization is facilitating the distribution of malware and the number of internet users is rising exponentially, we should expect more sophisticated attacks on the windows operating systems and ICT. Under these circumstances, we should review the windows operating system design strategy, focusing on security and build reliable operating system.

The open source phenomenon is definitely influencing in a positive way ICT and probably the trend will not change in the future. Open source projects are available to “millions of eyes” for scrutiny, improvement or testing. Nevertheless, it is likely that in the future will continue to experience the same issues mentioned above with some improvements in the area of standardization.

Some open source will definitely transform in commercial software provided that they have matured enough and captured a significant market share. Red Hat Linux for example, is a typical example of how open source software becomes commercial under the right circumstances.

CHAPTER TEN:

DISTRIBUTED OPERATING SYSTEM

INTRODUCTION

An operating system is a program that controls the resources of a computer and provides its user with an interface or a virtual machine that's more convenient to use than bear machine.

To begin with, we use the term distributed system to mean a distributed operating system as opposed to a database system or some distributed application system such as a banking system, another name for a distributed operating system is DIS-**CENTRALIZED OPERATING SYSTEM.**
Example of a centralized (not distributed) operating system are; MS-DOS, UNIX, and CP/M.

A distributed operating system is the one that look to its user like an ordinary centralized operating system but runs on multiple, independent, central processing unit (CPU). The key concept in distributed operating system is the TRANPARENCY. What determine a distributed operating system are the software and not the hardware.

In distributed system, the error can be made to tolerate both hardware and software error but it is the software error and not the hardware that cleans the error when it occurs.

Network OS is used to manage Networked computer systems and create, maintain and transfer files in that Network.
Distributed OS is also similar to Networked OS but in addition to it the platform on which it is running should have high configuration such as more capacity RAM, High speed Processor. The main difference between the DOS and the NOS is the transparent issue: Transparency:

- How aware are users of the fact that multiple computers are being used?

Types of Distributed Operating Systems

- Network Operating Systems
- Distributed Operating Systems

Network-Operating Systems

- Users are aware where resources are located
- Network OS is built on top of centralized OS.
- Handles interfacing and coordination between local OSs.
- Users are aware of multiplicity of machines.

Distributed-Operating Systems

- Designed to control and optimize operations and resources in distributed system.
  - Users are not aware of multiplicity of machines
  - Access to remote resources similar to access to local resources
  - Data Migration – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task
  - Computation Migration – transfer the computation, rather than the data, across the system
- Computation speedup – sub processes can run concurrently on different sites
  - Process Migration – execute an entire process, or parts of it, at different sites
  - Load balancing – distribute processes across network to even the workload
  - Hardware preference – process execution may require specialized processor
  - Software preference – required software may be available at only a particular site
  - Data access – run process remotely, rather than transfer all data locally

EXAMPLES OF DISTRIBUTED OPERATING SYSTEM

- The Cambridge Distributed Computing System
- Amoeba

- The V Kernel
- The Eden Project

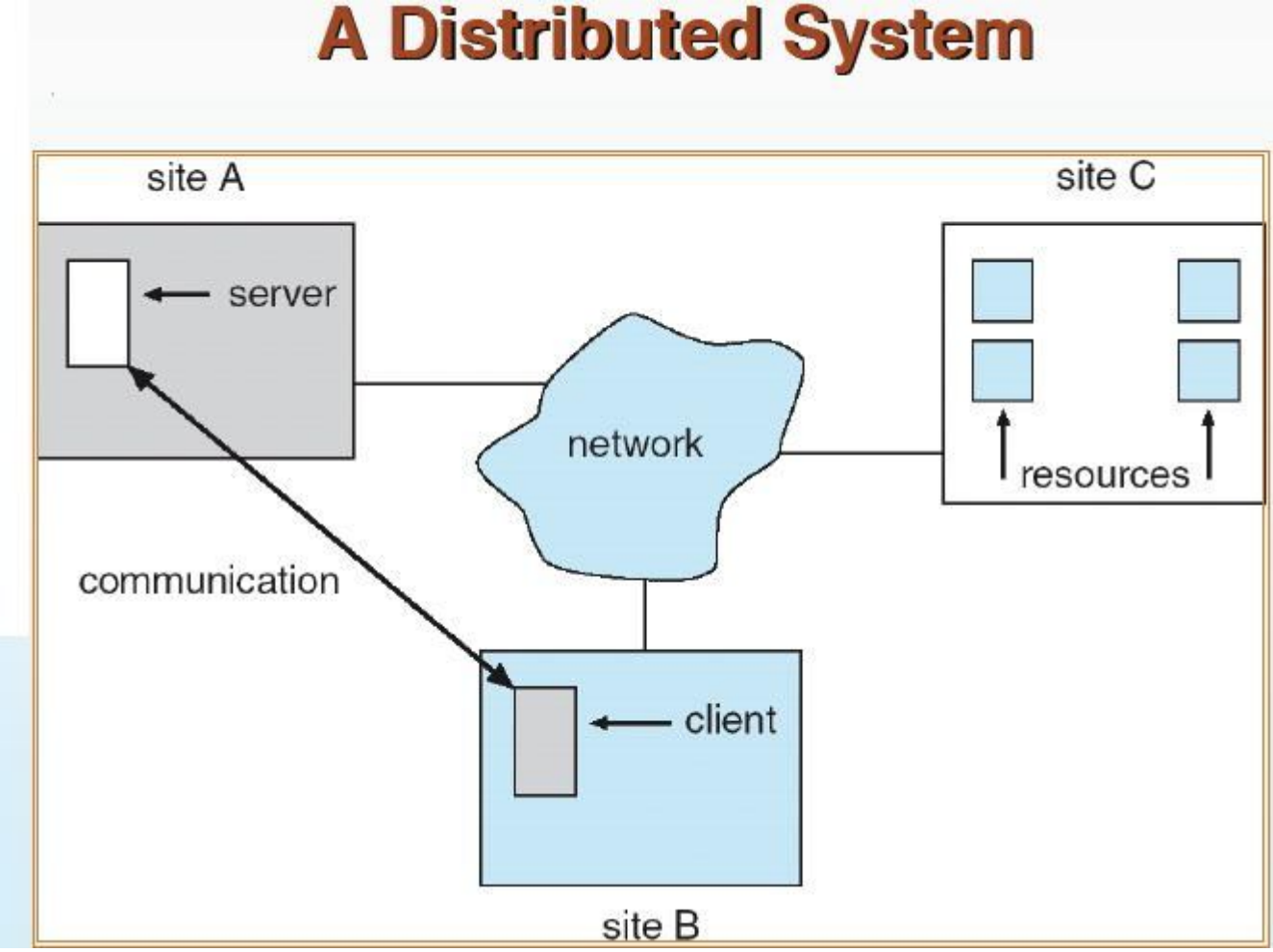
There are so many types of distributed operating system but these are chosen based on three criteria which are:

First, we only chose systems that were designed from scratch as-distributed systems (systems that gradually evolved by connecting together existing centralized systems or are multiprocessor versions of UNIX were excluded).

Second, we only chose systems that have actually been implemented; paper designs did not count.

Third, we only chose systems about which a reasonable amount of information was available.

NOTE: if a user can tell which computer he/she is using, then he/she is not using a distributed system. The user of a true distributed operating system should not know (or care) on which machine (or machines) their programs are running, where their files are stored and so on.



TRANSPARENCY

• Goal motivated by the desire to hide all irrelevant system-dependent details from the user, whenever possible.

• It is more important in distributed systems due to higher implementation complexities.

• **Shielding the system**-dependent information from the users is a trade-off between simplicity and effectiveness.

• **Access transparency** - Local and remote system entities must remain indistinguishable when viewed through the user interface. The distributed operating system maintains this perception through the exposure of a single access mechanism for a system entity, regardless of that entity being local or remote to the

user. Transparency dictates that any differences in methods of accessing any particular system entity—either local or remote—must be both invisible to, and undetectable by the user.

• **Location transparency** - Location transparency comprises two distinct sub-aspects of transparency, Naming transparency and User mobility. Naming transparency requires that nothing in the physical or logical references to any system entity should expose any indication of the entities location, or its local or remote relationship to the user. User mobility requires the consistent referencing of system entities, regardless of the system location from which the reference originates. Transparency dictates that the relative location of a system entity—either local or remote—must be both invisible to, and undetectable by the user.

• **Migration transparency** - Logical resources and physical processes migrated by the system, from one location to another in an attempt to maximize efficiency, reliability, availability, security, or whatever reason, should do so automatically controlled solely by the system. There are a myriad of possible reasons for migration; in any such event, the entire process of migration before, during, and after should occur without user knowledge or interaction. Transparency dictates that both the need for, and the execution of any system entity migration must be both invisible to, and undetectable by the user.

• **Concurrency transparency** - The distributed operating system allows for simultaneous use of system resources by multiple users and processes, which are kept completely unaware of the concurrent usage. Transparency dictates that both the necessity for concurrency and the multiplexed usage of system resources must be both invisible to, and undetectable by the user.

• **Replication transparency** - A system's elements or components may need to be copied to strategic remote points in the system in an effort to possibly increase efficiencies through better proximity, or provide for improved reliability through the duplication of a back-up. This duplication of a system entity and its subsequent movement to a remote system location may occur for any number of possible reasons; in any event, the entire process before, during, and after should occur without user knowledge or interaction. Transparency dictates that the necessity and execution of replication, as well as the existence of replicated entities throughout the system must be both invisible to, and undetectable by the user.

• **Parallelism transparency** - Arguably the most difficult aspect of transparency, and described by Tanenbaum as the "Holy grail" for distributed system designers. A system's parallel execution of a task among various processes throughout the system should occur without any required user knowledge or interaction. Transparency dictates that both the need for, and the execution of parallel processing must be both invisible to, and undetectable by the user.

• **Failure transparency** - In the event of a partial system failure, the system is responsible for the automatic, rapid, and accurate detection and orchestration of a remedy. These measures should exhibit minimal user imposition, and should initiate and execute without user knowledge or interaction. Transparency dictates that users and processes be exposed to absolute minimal imposition as a result of partial system failure; and any system-employed techniques of detection and recovery must be both invisible to, and undetectable by the user.

• **Performance transparency** - In any event where parts of the system experience significant delay or load imbalance, the system is responsible for the automatic, rapid, and accurate detection and orchestration of a remedy. These measures should exhibit minimal user imposition, and should initiate and execute without user knowledge or interaction. While reasonable and predictable performances are important goals in these situations, there should be no expressed or implied concepts of fairness or equality among affected users or processes. Transparency dictates that users and processes be exposed to absolute minimal imposition as a result of performance delay or load imbalance; and any system-employed techniques of detection and recovery must be both invisible to, and undetectable by the user.

• **Size transparency** - A system's geographic reach, number of nodes, level of node capability, or any changes therein should exists without any required user knowledge or interaction. Transparency dictates that system and node composition, quality, or changes to either must be both invisible to, and undetectable by the user.

• **Revision transparency** - System occasionally have need for system-software version changes and changes to internal implementation of system infrastructure. While a user may ultimately become aware of, or discover the availability of new system functions or services, their implementation should in no way be the prompt for this discovery. Transparency dictates that the implementation of system-software version changes and changes to internal system infrastructure must be

both invisible to, and undetectable by the user; except as revealed by administrators of the system.

SCHEDULLING TECHNIQUES

The hierarchical model provides a general model for resource control but does not provide any specific guidance on how to do scheduling. If each process uses an entire processor (i.e., no multiprogramming), and each process is independent of all the others, any process can be assigned to any processor at random. However, if it is common that several processes are working together and must communicate frequently with each other, as in UNIX pipelines or in cascaded (nested) remote procedure calls, then it is desirable to make sure that the whole group runs at once. Let us assume that each processor can handle up to N processes. If there are plenty of machines and N is reasonably large, the problem is not finding a free machine (i.e., a free slot in some process table), but something more subtle. The basic difficulty can be illustrated by an example in which processes A and B run on one machine and processes C and D run on another. Each machine is time shared in, say, 100-millisecond time slices, with A and C running in the even slices, and B and D running in the odd ones. Suppose that A sends many messages or makes many remote procedure calls to D. During time slice 0, A starts up and immediately calls D, which unfortunately is not running because it is now C's turn. After 100 milliseconds, process switching takes place, and D gets A's message, carries out the work, and quickly replies. Because B is now running, it will be another 100 milliseconds before A gets the reply and can proceed. The net result is one message exchange every 200 milliseconds. What is needed is a way to ensure that processes that communicate frequently run simultaneously. Although it is difficult to determine dynamically the inter-process communication patterns, in many cases a group of related processes will be started off together.

PROCESS MANAGEMENT

Process management provides policies and mechanisms for effective and efficient sharing of a system's distributed processing resources between that system's distributed processes. These policies and mechanisms support operations involving the allocation and de-allocation of processes and ports to processors, as well as provisions to run, suspend, migrate, halt, or resume execution of processes. While these distributed operating system resources and the operations on them can be either local or remote with respect to each other, the distributed operating system

must still maintain complete state of and synchronization over all processes in the system; and do so in a manner completely consistent from the user's unified system perspective.

As an example, load balancing is a common process management function. One consideration of load balancing is which process should be moved. The kernel may have several mechanisms, one of which might be priority-based choice. This mechanism in the kernel defines *what can be done*; in this case, choose a process based on some priority. The system management components would have policies implementing the decision making for this context. One of these policies would define what priority means, and how it is to be used to choose a process in this instance.

RESOURCES MANAGEMENT

Resource management in a distributed system differs from that in a centralized system in a fundamental way. Centralized systems always have tables that give complete and up-to-date status information about all the resources being managed; distributed systems do not. For example, the process manager in a traditional centralized operating system normally uses a “process table” with one entry per potential process. When a new process has to be started, it is simple enough to scan the whole table to see whether a slot is free. A distributed operating system, on the other hand, has a much harder job of finding out whether a processor is free, especially if the system designers have rejected the idea of having any central tables at all, for reasons of reliability. Furthermore, even if there is a central table, recent events on outlying processors may have made some table entries obsolete without the table manager knowing it. The problem of managing resources without having accurate global state information is very difficult.

PROCESSOR ALLOCATION

One of the key resources to be managed in a distributed system is the set of available processors. One approach that has been proposed for keeping tabs on a collection of processors is to organize them in a logical hierarchy independent of the physical structure of the network, as in MICROS. This approach organizes the machines like people in corporate, military, academic, and other real-world hierarchies. Some of the machines are workers and others are managers. For each group of k workers, one manager machine (the “department head”) is assigned the task of keeping track of who is busy and who is idle. If the system is large, there will be an unwieldy number of department heads; so some machines will function

as “deans,” riding herd on k department heads. If there are many deans, they too can be organized hierarchically, with a “big cheese” keeping tabs on k deans. This hierarchy can be extended ad infinitum, with the number of levels needed growing logarithmically with the number of workers. Since each processor need only maintain communication with one superior and k subordinates, the information stream is manageable. An obvious question is, “What happens when a department head, or worse yet, a big cheese, stops functioning (crashes)?” One answer is to promote one of the direct subordinates of the faulty manager to fill in for the boss. The choice of which one can either be made by the subordinates themselves, by the deceased’s peers, or in a more autocratic system, by the sick manager’s boss. To avoid having a single (vulnerable) manager at the top of the tree, one can truncate the tree at the top and have a committee as the ultimate authority. When a member of the ruling committee malfunctions, the remaining members promote someone one level down as a replacement. Although this scheme is not completely distributed, it is feasible and works well in practice. In particular, the system is self repairing, and can survive occasional crashes of both workers and managers without any long-term effects. In MICROS, the processors are monoprogrammed, so if a job requiring S processes suddenly appears, the system must allocate S processors for it. Jobs can be created at any level of the hierarchy. The strategy used is for each manager to keep track of approximately how many workers below it are available (possibly several levels below it). If it thinks that a sufficient number are available, it reserves some number R of them, where R 2 S, because the estimate of available workers may not be exact and some machines may be down. If the manager receiving the request thinks that it has too few processors available, it passes the request upward in the tree to its boss. If the boss cannot handle it either, the request continues propagating upward until it reaches a level that has enough available workers at its disposal. At that point, the manager splits the request into parts and parcels them out among the managers below it, which then do the same thing until the wave of scheduling requests hits bottom. At the bottom level, the processors are marked as “busy,” and the actual number of processors allocated is reported back up the tree. To make this strategy work well, R must be large enough so that the probability is high that enough workers will be found to handle the whole job. Otherwise, the request will have to move up one level in the tree and start all over, wasting considerable time and computing power. On the other hand, if R is too large, too many processors will be allocated, wasting computing capacity until word gets back to the top and they can be released. The whole situation is greatly complicated by the fact that requests for processors can be generated randomly anywhere in the system, so at any instant, multiple requests are likely to be in various stages of the allocation algorithm, potentially giving rise

to out-of-date estimates of available workers, race conditions, deadlocks, and more.

Failure Recovery

Failure Detection

Detecting hardware failure is difficult. To detect a link failure, a handshaking protocol can be used. Assume Site A and Site B has established a link. At fixed intervals, each site will exchange an I-am-up message indicating that they are up and running. If Site A does not receive a message within the fixed interval, it assumes either (a) the other site is not up or (b) the message was lost. Then, Site A can now send an "Are-you-up?" message to Site B. If Site A does not receive a reply, it can repeat the message or try an alternate route to Site B.
If Site A does not ultimately receive a reply from Site B, it concludes some type of failure has occurred in site B. Such failure could be that:
Types of failures:
- Site B is down
- The direct link between A and B is down
- The alternate link from A to B is down
- The message has been lost.

However, Site A cannot determine exactly **why** the failure has occurred

Reconfiguration

When Site A determines a failure has occurred, it must reconfigure the system:
1. If the link from A to B has failed, this must be broadcast to every site in the system
2. If a site has failed, every other site must also be notified indicating that the services offered by the failed site are no longer available.

When the link or the site becomes available again, this information must again be broadcast to all other sites.

Distributed Deadlock Detection

There are two kinds of potential deadlocks which are:

- resource deadlocks
- communication deadlocks

Resource deadlocks are traditional deadlocks, in which all of some set of processes are blocked waiting for resources held by other blocked processes. For example, if A holds X and B holds Y, and A wants Y and B wants X, a deadlock will result. In principle, this problem is the same in
Centralized and distributed systems, but it is harder to detect in the latter because there are no centralized tables.

The other kind of deadlock that can occur in a distributed system is a communication deadlock. Suppose A is waiting for a message from B and B is waiting for C and C is waiting for A. Then we have a deadlock. Chandy et al. [1983] present an algorithm for detecting (but not preventing) communication deadlocks. Very crudely summarized, they assume that each process that is blocked waiting for a message knows which process or processes might send the message. When a process logically blocks, they assume that it does not really block but instead sends a query message to each of the processes that might send it a real (data) message. If one of these processes is blocked, it sends query messages to the processes it is waiting for. If certain messages eventually come back to the original process, it can conclude that a deadlock exists. In effect, the algorithm is looking for a knot in a directed graph.

Redundancy Techniques

All the redundancy techniques that have emerged take advantage of the existence of multiple processors by duplicating critical processes on two or more machines. A particularly simple, but effective, technique is to provide every process with a backup process on a different processor. All processes communicate by message passing.

Whenever anyone sends a message to a process, it also sends the same message to the backup process. The system ensures that neither the primary nor the backup can continue running until it has been verified that both have correctly received the message. Thus, if one process crashes because of any hardware fault, the other one can continue. Furthermore, the remaining process can then clone itself, making a new backup to maintain the fault tolerance in the future. One disadvantage of duplicating every process is the extra processors required, but another, more subtle problem is that, if processes exchange messages at a high rate, a considerable amount of CPU time may go into keeping the processes synchronized at each message exchanged. If a process crashes, recovery is done by sending the most

recent checkpoint to an idle processor and telling it to start running. The recorder process then spoon feeds it all the messages that the original process received between the checkpoint and the crash. Messages sent by the newly restarted process are discarded. Once the new process has worked its way up to the point of crash, it begins sending and receiving messages normally, without help from the recording process.

STRENGTH AND WEAKNESS OF DISTRIBUTED OPERATING SYSTEM

STRENGTH

- The main goal of distributed system is the enormous rate of technological change in micro processor technology.
- Micro processors have become powerful and cheap compared with mainframes and minicomputer, so it has become attractive to think about designing large system that composes of many processors.
- Relative simplicity of software: each software has a dedicated function.
- Incremental growth.
- Reliability and availability.

WEAKNESS

- Unless one is very careful, it is easy for the communication protocol overhead to become a major source of in efficiency.
- With distributed systems, a high degree of fault tolerance is often, at least, an implicit goal.
- A more fundamental problem in distributed system is the lack of global state information.
- It is hard to schedule the processor optimally if you are not sure how many are up at the moment.

CONCLUSION

Distributed operating systems are still in an early phase of development, with many unanswered questions and relatively little agreement among workers in the field about how things should be done. Many experimental systems use the client-server model with some form of remote procedure call as the communication base, but there are also systems built on the connection model. Relatively little has been done on distributed naming, protection, and resource management, other than building straightforward name servers and process servers.

Fault tolerance is an up-and-coming area, with work progressing in redundancy techniques and atomic actions. Finally, a considerable amount of work has gone into the construction of file servers, print servers, and various other servers, but

here too there is much work to be done. The only conclusion that we draw is that distributed operating systems will be an interesting and fruitful area of research for a number of years to come.