ABHIYANTRIKI
An International Journal of Engineering & Technology
(AIJET)
Vol. 1, No. 1 (November, 2014) http://www.aijet.in/ eISSN: 2394-627X

Study of Security in Legendary Operating Systems

Surendranath Reddy Sreeyapureddy*
Department of Computer Science
Vikrama Simhapuri University

Abstract

As the computers advanced so much and it has more complexities, operating system responsibilities has increased, and give challenge to the operating system developers to build a secure operating system. Several operating systems were out in the market in which most of them are vulnerable. As the computer hardware is becoming more and more powerful, it is also vital to keep the software updated in order to utilize the hardware of the system efficiently and make it faster and smarter. The paper highlights some core issues that if dealt with in the operating system level would make use of the full potential of the computer hardware and provide an excellent user experience. Also in this paper we will see operating system security issues that computer industry has faced in desktop and mobile area and a brief idea is depicted to improve all operating systems for security.

Keywords*: Desktop Operating System, Mobile Operating System, Security, Android, Mac.*

****Author for correspondence*** sr1surendranathreddy@gmail.com

1. Introduction

Computer has improved marvelously in recent years; it has become necessity of everyone from just a scientific tool so that improving security is an issue to the users. For this purpose operating system plays a vital role in security of a system. Today we don't call a system by its manufacturer name but we call it by its operating system as MAC PC or Windows PC. Building a secure operating system has been and still is a major issue. Operating systems are becoming more dynamic day by day to utilize the full capacity of hardware's, as operating system is becoming more dynamic operating system faces some challenges which are still to be conquered. One of the major challenges is security of operating system. Informally, security is, keeping unauthorized entities from doing things you don't want them. This will help readers to have an overview of previous work

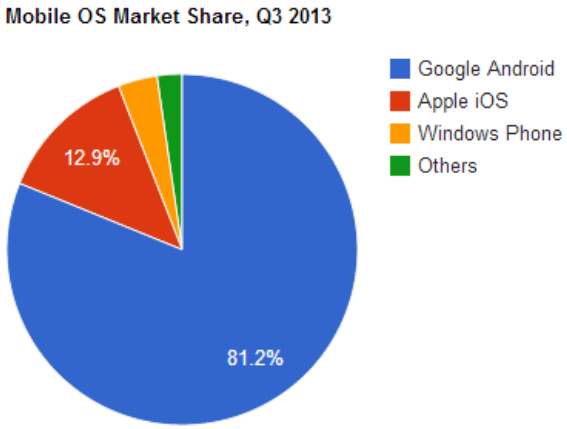

Figure 1

has been done for operating system security and give a direction to start their own study and will provide help for developers to keep these security issues in mind in development of operating system. We have a lot of different kinds of operating systems in the market but we will analyze most famous operating systems because these are used in large number publicly. Figure 1 and figure 2 are showing popularity chart which distinguish operating systems and show us the market share of operating systems in their respective area like desktop operating system and mobile operating system.

Security

Security has been and still remains a major concern for operating system developers and users alike. Informally security is keeping unauthorized entities from doing things you don't want them to do. It is operating system job to provide security against unauthorized users. Computer security is defined by three attributes, Confidentially, Integrity, Availability. Confidentially is prevention of unauthorized disclosure of information. Integrity is prevention of unauthorized modification of information, and Availability is the prevention of unauthorized withholding of information or resources. Operating system can provide sandbox, hashing password to protect against threats. In this paper some former researches techniques are used and we are going to explore more techniques and suggest the ideas for making a better security for operating system.

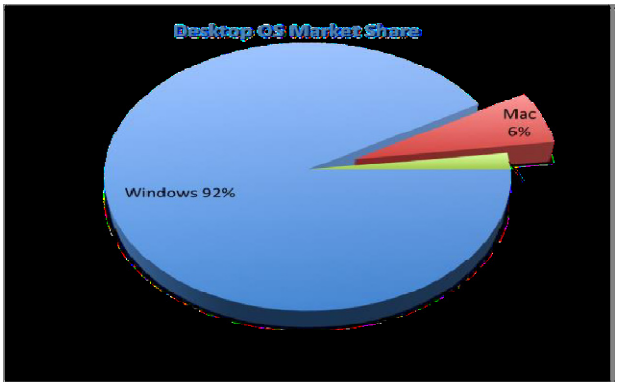

Figure 2

2. Review of Literature

Security has been improved recently but still there are flaws. Reasons are: a) most attacks are nowadays publicly announced and describes in detail on internet, b) vendor's attempt to offer backward compatibility which leaves open old weaknesses in the system.

Mobile Operating Systems

Mobile operating systems combination of personal computer and managing all hardware and optimizes. Many mobile operating systems can be seen in the market but two famous mobile systems now days are iPhone operating system (iOS) and Android.

a. iOS

iOS derived from OS X that share by darwing foundation [1]. iOS mainly used for the iPhone and iPad manufactured by Apple. There are 4 abstraction layers in iOS which are Core Services Layer, Media Layer, Core OS Layer and Cocoa Touch Layer. Core OS layer lies on the hardware and is the bottom layer. This layer function is to provide services including low level network access to external accessories and common operating system such as handling file system and memory management policy. Media layers contain audio, video and graphics geared towards creating the best multimedia experience on mobile device. Technology layer is for the ease of application builders that would look great. Cocoa Touch layer define the basic infrastructure and to support multitasking, touch based input, push notification and many high level notifications. This is the key framework for building iOS application.

b. Android

Android operating system is an open source and source code release under Apache license by Google. The OS is a linux based and the application software running on an application framework which includes Java compatible [2] libraries based on Apache Harmony.

Mobile Security Requirements

Here we are discussing 5 security requirements of mobile operating system which are:

a. Application Sandboxing

Sandboxing is mechanism for program to run separately, it is uses to limit the app boundary. When an app is build the permissions are assigned which cannot be changed dynamically on run time by an application or operating system kernel. Resources can be shared but these apps will never go beyond their defined limits which are declared at startup. In iOS all application shares same sandbox but in Android every application has its own sandbox.

b. Encryption

Encryption is the most effective method to secure data. Encryption is a technique to convert data into a secret code so data can be secure. For this purpose encryption algorithm is made and applied to data. You must have a secret key or password to decrypt data which is an encrypted file. When encryption is not applied to text it is simple plain text and when encryption is applied to data is called as cipher text. Encryption introduced to android in Android 3.0 Honeycomb version, first encryption method for Android is device encryption API which was released in Ice Cream Sandwich 4.0. Android applies encryption on disk level. Encryption is also applied to iOS which was introduced in iPhone 3GS version.

c. Memory Randomization

Memory randomization is a process where the memory application shared library and other in a device is located randomly. This technique is important to avoid attack on the memory of running application from any malicious code or virus. This technique is applied in iOS 4.3 version and later while in Android it is applied in Jelly Bean 4.1.1 and later versions.

d. Built-in Antivirus

In general, there are 3 types of popular malware that affects mobile such as Virus, Spyware and Trojan [3]. A virus is a malicious code which usually transmitted through email. Spyware is a program which collects the information about users without letting them know about it. Trojan is a desirable function but actually the purpose of the trojan is malicious. Android and iOS both were introduced with built-in antivirus features to avoid viruses, spyware, and trojans. Thousands of application can be downloaded from Google play safely because antivirus feature is not on android device but on Google Play [5]. It means any app downloaded except from Google Play can be very risky. We can download antivirus's applications from Google Play to avoid popular malware that affects our operating system. In iOS there is no need of antivirus because there is no room for virus to get into the system. In iOS there is only one place to download

application into the system that is App store, where every application is checked rigorously to make sure that it does not contain any kind of malicious code.

e. *Data Storage Format*
Disk Storage is a place where all the data is stored in build in storage or external storage. It is important to secure the storage to make sure your data is secured from any unwanted code. Commonly device has both internal and external storage. In Android, the data can be stored in both storages which in internal and external. Android implements standard crypto libraries to secure storage but it is as efficient as a password is applied. With the root access any unwanted code can access the files without any restriction and can spread malwares. While in iOS devices does not have external storage or memory but built-in storage. This requires permission to access the data. Data protection APIs in iOS are combined with passphrase which provide an additional layer of data protection. So iOS storage will be more secure than Android and make the application difficult to access the data from internal storage.

Table 1: A Quick Comparison of Android and iOS

Feature	Android	iOS
Sandbox	Every Application has its own Sandbox	All application shared same sandbox
Encryption	Encryption is on Disk Level	Encryption is on hardware level
Memory Randomization	Applied in 4.1.1 versions and later versions	Applied on 4.3 and later versions
Buit-in Antivirus	Antivirus can be downloaded from Google Play, virus checking is done on Google Play only, can be easily attacked because no built-in antivirus, any application outside from Google Play is risky.	No antivirus needed because application can be downloaded only from Apps store and checking been done in Apps store.
Data Storage	Have an external storage and it can be access by unwanted code.	No external storage which makes difficult for the unwanted code to access built-in storage.

By this comparison we conclude that iOS security is better than Android system.

Desktop Operating System
Many desktop operating system can be seen in market but here we are going to discuss three most famous and in most use desktop operating system which are Windows, Mac, and Linux.

[1] Windows

Microsoft windows are the most popular and most used operating system in the world. It's a graphical series of operating system of Microsoft. As figure 2 shown above MS Windows dominate 90% of desktop operating system shares. Microsoft Windows is a closed source operating system.

Windows Security

As the Microsoft windows is the most used operating system, it has more threats than other operating system as well. In 2005, over 1000 new viruses and worms were seen in six months duration, and 11000 malicious programs, viruses, trojans, back-doors and exploits were written for windows. Microsoft windows have released a lot versions and every operating system has some security issues.

User Space and Kernel Space

The Windows operating system is designed to support applications by moving more functionality into the operating system, and by more deeply integrating applications into the Windows kernel. Which doesn't have separation between user space and kernel space? Which may cause the critical damage to Kernel?

Update

Microsoft doesn't want to spend money on previous versions of windows, they don't provide windows update instead they are improving their flaws in upcoming versions.

Firewall

It only restricted inbound traffic and did not provide any mechanism for blocking or filtering traffic outbound from the Windows PC.

Hidden File Extensions

Windows continues to hide known file extensions by default. In other words, rather than displaying a full file name like "pcworld.docx", Windows will only display "pcworld". The idea is to make things more simple or user-friendly. We don't want to confuse the end-user with frivolous details like "docx", or "xls", or "mp3".

Internet Explorer

The security flaw allows attackers to slip malicious code into a website, using a compromised file. When a victim visits the tainted website using any of the Internet Explorer web browsers versions 6 through 11, attackers could gain full user rights over the victim's computer and potentially all information on it.

Adobe Flash Player

Gain access to a system and execute arbitrary code user privileges.

Memory

This problem was very common in Windows 9x family and Windows XP, although Windows XP has made a lot improvements over Windows 9x, but they both share this

memory problem, when any user program try to access the operating system memory or other user program it result come in memory dump and gets crashed.

[2] MAC OS

MAC OS is second most popular and widely used operating system which shares 6% of desktop operating system market share as shown in the figure 2. It is UNIX based graphical user interface operating system made only for MAC computers by Apple Inc.

MAC Security

MAC is second most popular operating system, so there are not too many viruses for MAC. But it doesn't mean MAC doesn't need security. Recently a trojan name variously Mac Protector, Mac Defender and Mac Guard showed on Apple machines, a window claiming to be the Apple Security Center pops up and indicate that virus has been found on this computer, and then it prompts to user to download Mac Protector and this software intended to steal credit card information [6].

When installing Mac OS X 10.5 Leopard, destination volumes may not appear in the installation window for a while, even though the volumes are visible while started from Mac OS X 10.4 or in Disk Utility. After performing an upgrade installation the default type of Mac OS X 10.5 Leopard, an administrator account may change to a standard one. After installing Mac OS X 10.5 Leopard on a 20-inch or 24-inch iMac (mid2007) computer (ones that have an aluminum frame), user may not be able to log in at login windows, login name and password are apparently accept but after a blue screen appears for a few seconds, the login windows reappears instead of your desktop. After installing Mac OS X 10.5 Leopard user may not be able to log in to account that has no password which was used in Mac 10.2.x and migrated to Leopard.

[3] Linux

In Linux, security system has two parts: first is authentication and second is access control. Some security issues regarding Linux operating system are as follows:

Local Security

Local users create a lot of problems for system. It is bad policy to provide accounts to people you don't know or for whom you have no contact information. It is better to follow some rules of thumb when offering access to your Linux machine: give users minimum privilege, monitor when and where they log in, remove inactive accounts and prohibit the creation of group user IDs.

Root Security

The root account has authority over the entire machine; you should use it only for specific tasks. Even a small mistake made while logging in as a root user can lead to significant problems. Follow the simple rules below and they will help you.

- When running complex commands, first run them in a non-destructive manner. A simple example is to do an "ls" before doing an "rm" so that you are sure about the files you are going to delete.

- Give users an interactive "rm" for deleting the files.
- Become "root only" to do specific tasks. If you want to experiment with something, go back to a normal user shell.
- The command path, which specifies the directories in which the shell searches for the programs, is very important. Limit the command path and never include "." (Signifying the current directory) in your command path.
- The /etc/security file contains a list of terminals that root can log in from. Be careful while adding an entry to this file.

File Security (Virtual File System)

Keep in mind the following points to help protect your systems and data stored on them. If you are exporting file systems using NFS, configure /etc/exports with the most restrictive access possible. Do not use any wild cards. Their integrity needs to be maintained, as they help in determining when and from where a user has entered your system. World-writable files can serve as a security hole. Also, world-writable directories are dangerous as they allow an intruder to add/delete files. You must locate the world-writable files on your system and make sure that you know why they are writable. It is also important to locate the un-owned files. The presence of un-owned files might also be an indication that an intruder has accessed your system. Before you change the permission on any system files, make sure you know what you are doing. Never make changes to the permission on a file just because it is the easy way to get things working.

File Permissions

Make sure that your system files are not open for casual editing by users and groups who do not have the appropriate permissions. The Linux operating system distinguishes the access control based on three characteristics: owner, group and other. Access to a file will be determined by permission bits and these bits are 'rwx' – where 'r' identifies 'read', 'w' identifies 'write' and 'x' identifies 'execute'. We can set or reset these three permission bits based on the kind of access that we are interested in giving to a user. This is considered as a basic level of preventing access to a file from unauthorized sources.

Integrity Checking

There is a very good mechanism to detect local attacks on your system. This is referred to as 'integrity checking'. Tripwire, Aide and Osiris are some of the popular integrity checkers. These integrity checkers will run a number of checksums on all important binaries and configuration files and compare them against a database of former, known values as a reference. Thus any changes in files can be easily flagged. Based on these signals, a system administrator can make appropriate changes so that integrity of important files is maintained.

Password Security

Most Linux distributions come with "passwd" programs that do not allow you to set a password that can be easily guessed. Thus, it is necessary to make sure that your "passwd" program is up to date. Linux uses a one-way encryption algorithm known as Data Encryption Standard (DES), which is used to encrypt your passwords. The encrypted password is stored in /etc/passwd. When you try to log in, the password you

type again gets encrypted and is compared with the entry in the file that stores your password. A match means you have entered the same password and you are given access to the system. Shadow passwords are a means of keeping your encrypted password information secret from the normal users. Recent versions of both Red Hat and Debian Linux use shadow passwords by default. Shadow passwords are saved in /etc/shadow and they can be read only by privileged users.

Kernel Security
As the kernel controls your machine's networking, it is essential to keep it secure. Let's look at some popular kernel configuration options that relate to security. IP forwarding: If you enable IP forwarding, your Linux box becomes a router. You can enable or disable IP forwarding by using these commands:

```
root# echo 1 > /proc/sys/net/ipv4/ip_forward /* for enabling */
root# echo 0 > /proc/sys/net/ipv4/ip_forward /* for disabling */
```

IP Firewalling
This option is very useful if you want to protect your dial-up workstation from someone entering via your PPP dial-up interface.

IP Firewall Packet Logging
This option displays the information about the packets your firewall receives.

Other Security Implementations
The one to consider here is the implementation of IPSEC for Linux. IPSEC is a mechanism to create cryptographically secure communications at the IP network level. The main idea here is to provide authentication, integrity, access control and confidentiality for your information.

Security Guidelines
Among all the concerns surrounding the writing of good code, security necessarily comes at the top. Security problems can come from people actively trying to penetrate your security or from simple issues such as someone providing unexpected inputs to a program or running some wrong commands. Too much access to systems can mean that users – even with legitimate access – can cause trouble, either accidentally or on purpose.

3. Objectives & Methodology

Security has been and still remains a major concern for operating system developers and users alike. Informally speaking, security is, keeping unauthorized entities from doing things you don't want them to do. Operating system protection involves protection against unauthorized users as well as protection of file systems. File permissions are based on user identity, which in turn are based on user identity, which in turn are based on authentication. Hence authentication of users has to be highly secure such that any unauthorized user doesn't hack in along with proper mechanism to let in genuine users. Various authentication mechanisms have been and are being used in operating systems, like the old-fashioned password authentication, where a plaintext

password is stored. This mechanism has been proven to be easily hackable, so another technique that provides an alternative is Hashed Passwords.

General Algorithm

Store f(Pw), where f is not invertible
When user enters Pw, calculate f(Pw) and compare.

Attackers can still use password-guessing algorithms; therefore most operating systems use access control mechanisms to protect the hashed passwords. Another authentication mechanism used is the Challenge/Response Authentication. Here what happens is the server knows Pw and sends a random number N, both sides then calculate f(Pw,N) where f is some encryption algorithm, although it must be noted that this mechanism is not very famous with operating systems. The reason being that, even in this case a person who guesses N or finds it out and comes to know f(Pw,N) can run password-guessing algorithms, so it is not that very different from the hashed-password authentication in terms of security. These days use of biometrics has become a major user authentication mechanism. Such techniques include fingerprint readers, iris scanner, etc. Although biometrics works fine if used locally, yet even these methods are susceptible to spoofing attacks. Hence we can infer that even the best and the most hi-tech authentication has its limitations.

When talking about operating system security, authentication attacks will make the bottom of priority list. The major problems are attacks like, Trojan Horses, Login spoofing and Buggy Software. Trojan Horses are basically programs that are disguised programs, meant to harm the system and its resources. Someone may be tricked into running a program that may adversely affect that user; his system or data. Although Linux, UNIX and other Unix-like operating systems are generally regarded as very protected, yet they are not immune to computer viruses. For example, consider a virus program written in C, which goes on creating new files and allocating space in an infinite loop! Will Linux be safe in that case? Hence viruses are a threat to all operating systems. Although it must be noted that there has not yet been a widespread Linux malware (malware as in any malicious software) threat of the type that Microsoft Windows software face; this is mostly because of the following reasons: the user base of the Linux operating system is smaller compared to Windows; malwares' lack root access; and fast updates for most Linux vulnerabilities.

Operating systems may use the following mechanisms to avoid attacks of this type:

- Operating systems can provide sandboxes: Sandboxes are environments where a program can execute but should not affect the rest of the machine.
- The trick here is, permitting limited interaction with outside while still providing the full functionality of the operating system. Or in other words, the file system can be kept out of unauthorized access and 3rd party softwares may be allowed minimum access to file-systems.

Race conditions can also be a critical security issue. To illustrate such a situation, consider a privileged program that checks if a file is readable and then tries to open it as root. The attacker passes it a symbolic link, in the interval between the two operations; the attacker removes the link and replaces it with a link to a protected file. This would give him direct access to the

protected file area and into the system. So here, an attacker takes advantage of the race condition between two operations to get access into the protected area of the operating system. The only way to overcome such attacks is to provide only atomic operations to access files and strict restrictions on their access by other users other than root.

Security is not only an issue with the operating systems in desktops and laptops; the operating systems of tablets and cell-phones also have the same security issues but these issues in phones are the most critical because if an attacker gets into the operating system of a phone, the attacker may get access to the personal data (viz. contacts, messages, etc) of the victim; and moreover the user base of these smaller devices like smart-phones and tablets in increasing at an alarming rate and the amount of data sharing between these devices is far more than that between computers.

Management

Managing the system memory is a very important function of an operating system. Hence the success of any operating system also depends to some extent on how well the operating system manages the system memory. There have been numerous mechanisms that have been researched upon and implemented in this area of operating system development. Today, an operating system has to execute tasks on a huge amount of data but in the early days the catch was that to operate on data, it had to be present in the primary memory and primary memory cannot be as much as the secondary memory. So the researchers and developers started finding alternate ways of storage and execution of data. During this time came a concept called paging. In operating systems, paging is one of the memory management schemes by which the system can store and retrieve data from the secondary storage for use in the main memory. In this scheme, the operating system retrieves data from secondary storage in same size blocks called pages. The main function of paging is performed when a program tries to access pages that are not currently mapped to the RAM. This situation is known as a page fault.

Objectives

- Determine the location of data in auxiliary storage.
- Obtain an empty page frame in RAM to use as a container for data.
- Load the requested data into the available page frame.
- Update the page table to show the new data.
- Return control to the program, transparently retrying the instruction that caused the page fault.

Until there is not enough RAM to store all the data needed, the process of obtaining an empty page frame does not involve removing another page from RAM. If all page frames are non-empty, obtaining an empty page frame requires choosing a page frame containing data to empty. If the data in that page frame has been modified since it was read into RAM, it must be written back to its location in secondary storage before being freed; otherwise, the contents of the page's page frame in RAM are the same as the contents of the page in secondary storage, so it does not need to be written back to secondary storage. If a reference is then made to that page, a page fault will occur, and an empty page frame must be obtained and the contents of the page in secondary storage again read into that page frame. Efficient paging systems must determine the page frame to empty by choosing one that is least likely to be needed within a short time. There are various page replacement algorithms that try to do this. Most operating systems use some

approximation of the least recently used (LRU) page replacement algorithm (the LRU itself cannot be implemented on the current hardware) or a working set-based algorithm.

Paging is a very important feature for memory management and is made use of by most of the commercially available operating systems. For example, consider paging in Windows. Almost all memories in windows can be paged out to disks. This is where page file comes into play; it's where most pages are placed when they are not resident in the physical memory. However, not everything gets written into page files, they get written to specific mapped files. Better than that, the pages only get written if they have been modified. If they have not been altered since they were read from the file, windows doesn't have to write the pages back out; it can just discard them. If it ever needs the pages again, they can be safely re-read from the files. Although paging is a very efficient mechanism yet challenges still exist in this area, that need to be overcome if the performance of the system has to be increased. Operating systems today have taken paging to the next level, by allowing sharing of pages between different processes. This technique has an important advantage, that is, it avoids duplication of pages for multiple processes. Or in other words, if pages were not shared between processes, then each process would have had to acquire its own copy of a page that is being used by another process. Hence by allowing sharing of pages, the execution time of instructions goes down, in turn making the operating system run faster. This memory sharing is useful, especially in low-memory systems, but the current technique present for sharing of pages has its limitations; major one being that the operating system only shares memory that corresponds to memory mapped files. That is because this is the only time that the operating system knows that pages are identical. For regular data there is no page sharing.

A new scheme for page sharing is going to be implemented by vendors. Here, the system will periodically scan memory, and when it finds two pages that are identical, it will share them, reducing the memory usage. If a process then tries to modify the shared page, it will be given its own private copy, ending the sharing. This mechanism will have a huge effect on virtualization. When virtualizing, the same operating system may be running multiple times, meaning that the same executable files are loaded several times over. So the traditional memory-mapped file approach to memory sharing cannot kick in here. Each virtual operating system is loading its own files from its own disk image. This is where memory de duplication is useful; it can see that the pages are all identical, and hence it can allow sharing even between virtual machines. This is another technique that is used by some operating systems (Mac OS X) for memory management. As per this method, when the operating system needs memory it will push something that isn't currently being used into a swap file for temporary storage. When it needs access to that data again, it will read the data from the swap file and back into memory. In a sense, this can create unlimited memory, but it is significantly slower since it is limited by the speed of the hard disk, versus the near immediacy of reading data from RAM. Even this mechanism has a flaw. For example, consider that processes A, B, C are to be executed one after the other wherein A and C need same resources but B needs totally different resources. Another assumption here is that there is no memory left in the RAM. So here once process A is finished, process B will have to run, but since B needs different resources and resources of A are not required anymore for now, they are shifted into swap file and resources for B are loaded in place of that. Now when C is to be executed, again the resources that had been shifted to swap file has to be shifted back to the

RAM. So here we see how redundant swapping of data takes place and these results in slow processing speed.

Methodology
The operating systems today use some approximation of the LRU (least recently used) algorithm as the LRU itself has not been completely implemented on any present machine. To increase responsiveness, paging systems must employ better strategies to predict which page will be needed soon. Such systems will attempt to load pages into main memory preemptively, before a program references them. Operating systems will need better methods of page sharing, such that page sharing for regular data and not only for memory-mapped data can be achieved. If swapping mechanism is to be used for memory management, then proper measures need to be taken to avoid redundant sharing of data as much as possible.

4. Theoretical Perspective

Nowadays usage of more than one processor in a computing system has become a common occurrence. Operating systems should have efficient mechanism to support more than one processors and the ability to schedule tasks between them. There are many variants of this basic theme and the definition of multiprocessing may vary with context.

In a multiprocessing system, all CPUs may be equal, or some may be reserved for special purposes. A combination of hardware and OS software design considerations determine the symmetry or lack of it in a given system. For example, hardware or software considerations may require that only one CPU respond to all hardware interrupts, whereas all other work in the system may be distributed equally among CPUs; or execution of kernel-mode code may be restricted to only one processor at a time whereas user-mode code may be executed in any combination of processors. Multiprocessing systems are often easier to design if such restrictions are imposed, but they tend to be less efficient than systems in which all CPUs are utilized. Systems that treat all CPUs equally are called Symmetric Multiprocessing Systems (SMP). In systems where CPUs are not equal, system resources may be divided in a number of ways including Asymmetric Multiprocessing Systems (ASMP), Non-Uniform Memory Access (NUMA) multiprocessing systems and Clustered Multiprocessing Systems. In computing, SMP involves a multiprocessor computer architecture where two or more identical processors can connect to a single shared main memory. Most common multiprocessor systems today use SMP architecture. In case of multi-core processors, the SMP architecture applies to the cores, treating them as separate processors. SMP systems allow any processor to work on any task no matter where the data for that task is located in the memory. With proper OS support SMP systems can easily move tasks between processes to balance the workload efficiently.

Asymmetric multiprocessing varies greatly from the standard processing model that we see in the personal computers today. Due to the complexity and unique nature of this architecture it was not adopted by many vendors during a brief stint. While SMP treats all of the processing elements in the system identically, an ASMP system assigns certain tasks only to certain processors. Although hardware level ASMP may not be in use, the idea and logical process is still commonly used in applications that are multiprocessor intensive. Unlike SMP applications which run there threads on multiple processors, ASMP application will run on one processor but outsource smaller tasks to other processors. The operating systems may also make use of ASMP

architecture for critical tasks like the tasks that may make use of system files. Operating systems can dedicate one processor called the Master processor to implementation of tasks required on the system files while smaller related tasks may be delegated to other processors called the Slave Processors. Although the basic architecture will still be SMP yet for critical tasks the ASMP architecture may be used. Modern CPUs operate considerably faster than the main memory they use. In the early days of computing and data processing the CPU generally ran slower than its memory. The performance lines crossed in the 1960s with the advent of high speed computing. Since then, CPUs increasingly "starved for data" have had to stall while they wait for memory accesses to complete. Limiting the amount of memory access provides the key to extracting high performance from a modern day computer. For commodity processors this means installing an ever increasing amount of high speed cache memory and very sophisticated algorithm to avoid cache misses. But dramatic increases in size of the operating systems make the problem considerably worse. Now a system can starve several processors at the same time, notably because only one processor can access memory at a time. NUMA attempts to address this problem by providing separate memory for each processor, avoiding performance hit when several processors attempt to address the same memory. Of course, not all data ends up confined to a single task, which means that more than one processor may require the same data. To handle these cases, NUMA systems include additional hardware or software to move data between banks. This operation slows the processors attached to those banks, so the overall speed increase due to NUMA depends heavily on the exact nature of tasks that are running. This architecture can substantially increase the performance but for that there has to be proper hardware and the operating system must provide some mechanism to efficiently schedule the access to multiple processor memory. If NUMA architecture is implemented successfully both in the hardware and in the OS level then it could go a long way in speeding up processing with multiple processors.

5. Result & Discussion

Following are result of the work discussed above:

- Secure operating systems make it easier to write secure applications.
- There is a need for more flexible permission model. The models present today are either too simple or too restrictive.
- No commercial operating system is secure enough.
- There will always be buggy code, but the trick is to build an application and an operating system that will mostly restrict attacks and will protect the important assets of the system.
- Although most of the operating systems today use SMP architecture yet with proper operating system support SMP systems can move tasks between processors more freely and thus balance the workload effectively.
- Operating Systems can implement a hybrid of SMP and ASMP architectures wherein, while all the tasks can be delegated using SMP architecture, the tasks that make use of system files can make use of ASMP architecture to implement that part.
- NUMA architecture can be seriously looked upon during future operating system design such that a way to integrate this architecture into the system is reached. If this happens, it could go a long way in speeding up the processing with multiple processors.

6. Conclusion

As the user awareness of technology is increasing, so, there is expectation. Hence although operating systems have progressed a lot, yet still there is a lot of ground to cover in this field.

Operating systems research is a very vast field and the reason for this is mostly because the hardware is becoming stronger and faster by the day and hence there is a race for the operating systems to keep up. The key issues pointed out in this paper if addressed, will make our computation even more wonderful than the present.

References

[1] Ahmad, M. S., Musa, N. E., Nadarajah, R., Hassan, R., & Othman, N. E. (2013). Comparison between Android and iOS operating system in terms of security. *In*: Proceedings of Information Technology in Asia (CITA), 2013 (1-4).

[2] Khadijah, Wan Mohd., Ghazali, R. H., & Zulkarnain, M. A. (2013). A network device simulator. *In*: Proceedings of IEEE ICACT 2013, Pyong Chang, Korea (pp.378-381).

[3] Qing, L., & Clark, G. (2013). Mobile security: a look ahead. *IEEE Transactions on Security & Privacy*. *11*(1), 78-81. doi: 10.1109/MSP.2013.15

[4] http://i0.wp.com/blog.goyello.com/wp-content/uploads/2014/01/2014-01-23-13_15_18.png

[5] http://www.laridian.com/images/2013-04%20Desktop%20OS%20Market%20Share.png

[6] http://computer.howstuffworks.com/macs/10-differences-between-macs-and-pcs.htm#page=9

[7] Oldberg, R. P. G. (June, 1974). Survey of virtual machine research. *IEEE Computer Magazine*. pp. 34–45.

[8] Bhargava, R. S., Erebrin, B. S., Padini, F., & Manne, S. (2008). Accelerating two-dimensional page walks for virtualized systems. *In*: Proceedings of 13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).

[9] Ben-Yehuda, M., Mason, J., Xenidis, J., Krieger, O., Van Doorn, L., Nakajima, J., Mallick, A., & Wahlig, E. (2006). Utilizing IOMMUs for virtualization in Linux and Xen. *In*: Proceedings of Ottawa Linux Symposium (pp. 71–86).

[10] Levasseur, J., Uhlig, V., Stoess, J., & Gotz, S. (2004). Unmodified device driver reuse and improved system dependability via virtual machines. *In*: Proceedings of 6th Symposium on Operating Systems Design & Implementation (p. 2).