ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

DIPLOMA IN CYBER SECURITY

DCS-04 APPLICATION CYBER SECURITY

BLOCK

1 SYSTEM SECURITY

Unit-1	Desktop Security
Unit-2	Programming Bugs and Malicious Codes
Unit-3	Database Security
Unit-4	Operating System Security

EXPERT COMMITTEE

Dr. P.K. Behera (Chairman)
Reader in Computer Science
Utkal University
Bhubaneswar, Odisha

Dr. J.R. Mohanty (Member)
Professor and HOD
KIIT University
Bhubaneswar, Odisha

Sri Pabitrananda Pattnaik (Member)
Scientist-E, NIC
Bhubaneswar, Odisha

Sri Malaya Kumar Das (Member)
Scientist-E, NIC
Bhubaneswar, Odisha

Dr. Bhagirathi Nayak (Member)
Professor and Head (IT & System)
Sri Sri University
Bhubaneswar,Odisha

Dr. Manoranjan Pradhan (Member)
Professor and Head (IT & System)
G.I.T.A
Bhubaneswar, Odisha

Sri Chandrakant Mallick (Convener)
Consultant (Academic)
School of Computer and Information Science
Odisha State Open University
Sambalpur, Odisha

DIPLOMA IN CYBER SECURITY

Course Writer

Chandrakant Mallick
Consultant (Academic)
School of Computer and Information Science
Odisha State Open University, Sambalpur

Course Writer

Bijay Kumar Paikray
Lecturer, School of Computer Science and Engineering,
Centurion University of Technology and Management, Odisha

UNIT-1 DESKTOP SECURITY

Unit Structure

1.0 Introduction

Use of desktop still dominates the other computing devices in many organizations. But the desktops are more vulnerable to security threats and attacks through intrusions. So it is crucial to have a desktop security policy by an organization. Otherwise the desktops will serve as the gateways to access the organization's highly valuable and confidential information assets. This will be a potential threat to information security in an organization.

For this reason, we should really make sure that the basic principles of information security –confidentiality, integrity and availability – is strictly maintained.

In this unit we will discuss different ways to ensure security of desktop computers as well as laptops and notebooks in a network computing environment by enforcing security policies and technology within an organization. We may often refer this as Computer Security as a whole.

1.1 Learning Objectives

After learning this unit you should be able to

- Define a desktop computer
- Understand the meaning of securing desktops.
- Know the need for desktops security.
- Know the desktop security policies.
- Remember the password policies for security.

1.2 Overview of Computer Security

Maintaining Computer Security is not as simple as it may appear to the novice. It is not a one-time affair rather a continuous process.

Computer Security is a "chess game" between the attacker and the security administrator. The attacker only needs to find a single vulnerability to penetrate the system, while the administrator needs to patch all holes to ensure Computer Security.

It is a natural tendency to disregard security problems until a security failure occurs. But ensuring Computer Security is a process. It requires a constant monitoring and a long-term perspective. It may be often an afterthought – added after the system has been designed. Some users think security is restricting them in their job. However, it is very important to ensure security to the computer system, its data, databases as well as its connectivity.

We cannot add security but only reduce insecurity. A computer system is never 100% secured. Consider the threats and the value of what you protect. Computer items must be protected only until they lose their value. They must be protected to a degree consistent with their value.

1.3 What is a Desktop Computer?

A **desktop computer** is a personal computer designed for regular use at a single location or near a desk or table due to its size and power requirements. The most common configuration has a case that encloses the power supply, motherboard (a printed circuit board with a microprocessor as the central processing unit (CPU), memory, bus, and other electronic components), disk storage (usually one or more hard disk drives, optical disc drives), a keyboard and a mouse for input, a monitor and additionally a printer for output.

An all-in-one desktop computer typically combines the case and monitor in one unit. On desktop security, it may state that all unattended IT equipment's must have appropriate and approved security protection.

1.4 Why Desktops need to be secured?

Many network security intrusions occur due to lack of knowledge and best security practice among employees in an organization. So, desktop systems face a number of vulnerabilities and security threats.

Desktop is the entry point to the organization's information resources. If the security of the desktop is weak, potential intruders can easily by-pass the first obstacle.

Some of the threats that need to be constantly communicated to desktop users are:

- Using programs that send password to the intended person.
- Bad password management, weak password, sharing password, never change password, "post-it note" habits etc.
- Guest accounts or open accounts.
- Social engineering attacks.
- Virus and other malicious code of attacks.
- Unsolicited email attachments.
- Downloading software from un-trusted Internet sites.
- Installing software from un-trusted sources.
- Modem connection to desktop within the LAN thus creating a backdoor.
- Unattended desktop, without screen lock.
- Packet sniffing.
- Bad desktop management, no anti-virus, outdated virus signature, no backups, no desktop lock, opens folder shares without password, opening insecure access etc.

1.5 What do you mean by Securing Desktops?

Securing desktops within an organization means the following.

- Protection of the desktop software and its configuration to allow for its continued use.
- Protection of the local desktop, supporting files and configuration.
- The control of employee's behavior on the desktop and the protection of content.
- Ensuring confidentiality, integrity and availability of sensitive data and databases in an organization.

1.6 Desktop Security Policies

A basic list of the security policies would include an anti-virus program, use of licensed and Open source Software, Software updates, Anti-spyware, a personal firewall, file-encryption software etc. as security solutions as follows.
Each one of these mechanisms will help ensure at least one or more of the three basic principles of security, Confidentiality, Integrity and Availability are addressed properly. Let us discuss how each of these mechanisms fit into the overall desktop security policy.

1. **Anti-Virus:** Install and maintain current anti-virus software. Check for and install any updates to both the software and virus definitions on a regular basis. This software will check incoming data for viruses, scan your computer for existing viruses, and make sure no one is installing data-collecting software on your computer without your knowledge.

2. **Licensed and Open Source Software**: All the software you use for work should also be from reputable sources with valid licensing and regularly updated. Or otherwise use Open source software. Pirate software can contain malware (as well as being illegal). Setting your software to auto-update means it will always be protected against the latest threats.

3. **Software updates**: Regularly check for and ensure that software updates/patches are installed. This includes, but is not limited to, operating system updates, application patches and firmware updates.

4. **Anti-spyware Software:** A computer program that attempts to identify and remove spyware from a protected computer. This software can be classified as part of the Confidentiality category. Symantec Anti-Virus now includes this function or there is freeware such as Spybot Search and Destroy. Microsoft is now offering its own version called Windows Defender.

5. **File-Encryption Software:** Computer programs that will take data stored on a media device and apply an encryption algorithm to it in order to render the data unreadable to those without proper access. This software would be part of the Confidentiality category. An example of this software is Credant

Technologies Mobile Guardian. Microsoft includes this as an option on its Active Directory enabled operating systems.

6. **Implement Physical Security:** Measures Workstations must be configured to require a password to access the system. Enable screen locking features to prevent unauthorized access to one's machine while not in use. Any exceptions such as public terminals, kiosks, or lab computers should be documented. This would be classified in the Confidentiality and Integrity categories.

7. **Disable Unnecessary Services:** Many operating systems may be configured (by default) to permit access to ones system from other computers on a network. An assessment should be performed to identify all services enabled on a system. Any unnecessary services should be disabled and any exceptions (services left enabled due to business or operational requirements) documented.

8. **Limit Use of Privileged Accounts:** Under certain circumstances normal users may be issued system accounts that have administrative or privileged access to a system. Users should limit the use of these accounts (to the specific tasks requiring them) and not use them for general work purposes.

9. **Host-based Firewall Software:** Host-based firewalls help protects individual systems from malicious attacks initiated by other systems on a computer network. Workstations are required to have locally installed firewall software and have it configured in a secure manner approved by the UA Chief Security Officer.

10. **OS Patch Management Software:** There are multiple variations of patch management software available. Some require a managed environment to accomplish patching a desktop where others allow the desktop to update themselves on their own schedule. Either way, it is the means that an operating system receives minor fixes (patches) to fix bugs or security vulnerabilities in the software. This last one may be classified as part of the Integrity and Availability groups. Many different examples are available for this to include Microsoft's Systems Management Server, and Microsoft's Windows Server Update Services.

1.7 Best Practices for Desktop Security

All computer operating systems have vulnerabilities are subject to security risks. In a networked environment, such as a college campus, a compromised computer can affect other computers and disrupt services throughout the campus, personal information can be compromised leading to identity theft and intellectual property can be stolen. In order to reduce the risk of a successful intrusion and to minimize

the damages, some basic steps and procedures are to be followed to minimize security exposures and resulting disruptions.

1.7.1 Basic steps to ensure Desktop Security

The basic steps for securing desktop systems are as follows:

- Keep operating system patches up to date.
- Use encryption to securely encode sensitive information
- Install antivirus software; configure for daily updates
- Install and configure a personal firewall
- Keep application and software patches up to date (e.g., Microsoft Office, browsers, etc.)
- Follow best practices when opening email attachments
- Follow secure password policies
- Follow best practices for user account security
- Eliminate unnecessary network services, applications, and processes
- Avoid peer-to-peer file sharing
- Install and configure anti---Spyware programs
- Configure system restore points to protect your current configuration
- Perform regularly scheduled backups to protect data
- Turn off computer when not in use; restrict physical access to computer.

The simple way to desktop security from physical access is to either to set a password protected screen saver or, even better, to get into the habit of locking computers when users walk away from them. A screen saver timeout should be utilized so if a user walks away from their computer, the password-protected screen saver would come up. Personally, I have mine set to 15 minutes, but upon doing a Google search regarding typical screen saver timeouts, the average response was 10 minutes. This information can and should be supplemented with information on how to do this. Tactics a potential attacker could utilize (e.g. Shoulder surfing, key loggers, etc) also need to be addressed. Another item that could be addressed is to make sure users understand that it is important that they shut down their computers at the end of the day. Sometimes this allows for valuable updates to be applied and doing your own part for a greener environment. If somehow a potential attacker gains access to a computer that is turned off, they will be less likely to utilize it than one that is already turned on and unlocked.

1.7.2 Patching the Operating System

One of the most important and fundamental routine tasks to perform is to patch your computer on a regular basis.

All operating systems have “holes” in them. These “holes” are vulnerabilities in the software code that can be exploited to gain access to the computer system. There are programs on the Internet that are constantly searching for unpatched

computers. Prior to connecting a computer to the Internet, please be sure that the latest critical patches are installed.
Both Microsoft and Apple have provided programs that will automatically check for updates on a regular basis. The settings should be configured to check for updates daily.
To confirm or configure the proper settings follow these steps:

WINDOWS-7

- Open Windows Update by clicking the Start button. In the search box, type Update, and then, in the list of results, click Windows Update.
- In the left pane, click Change settings.
- Under Important updates, choose the option "Install updates automatically (Recommended)"
- Under Recommended updates, Select the Give me recommended updates the same way receive important updates checkbox, and then click OK.
  If you' re-prompted for an administrator password or confirmation, type the password or Provide confirmation.

1.7.3 Patching applications and software

Just as operating systems require patches to correct security flaws, so do applications. Please make sure that your application have been patched and are up to date. In particular, pay close attention to the browser(s) you use (e.g., Internet Explorer, Mozilla Firefox, Apple's Safari, Netscape, etc.). These programs are often the object of attack. Similarly, other applications are vulnerable if not patched appropriately.
Patching is also required in order for the applications to operate properly with the operating system as it changes over time due to patching.
In order to update Microsoft Office for Windows or Macintosh open one of the programs (Word, Excel, Access, and Power Point), click the Help button, and select Check for Updates.

1.7.4 Steps for basic Linux Desktop Security

Linux is less vulnerable than Windows, but that doesn't make it immune to attackers. It's not always about security flaws, buffer overflows or denial of service attacks. Most intruders exploit incorrect system configurations or access permissions which are often caused by user ignorance. Remember that the basic rules that should reduce the security risk include the following.

1. **Download** the ISO for your preferred distro **from trusted sources**. It's recommended to visit the official web page and select a download method from there. If you are downloading from unofficial torrent sites for higher speed rates, make sure they're using the same tracker. Upon downloading always check the SHA1/MD5 sum.

2. **Don't perform a full install**. Select only packages that you need, why waste the disk space? Fewer packages mean fewer bugs.

3. After the installation, **disable any unwanted services**. A running service means an open port to the outside. If you don't need that service, it's better to disable it. Run **netstat -ntlp | grep LISTEN** as root to find out which services are running. Also, if you don't use IPv6, you can safely deactivate IPv6 support in your network card configuration.

4. **Run a firewall**. Either you use a distro specific GUI or configure it yourself, the firewall is a must-have security measure if you have an active network connection, as it drops unnecessary traffic and blocks a possible intruder.

5. **Configure tcp_wrappers**. It's really easy to do it and it gives you an extra layer of security. You can control access to all services (e.g. SSH) that are linked against libwrap or run by a super daemon (e.g. xinetd).

6. **Avoid using the root account**. Configure sudo access for your user, it's safer.

7. **Update your system** on a regular basis. Don't mind the daily updates, they're meant to resolve bugs and keep your machine more secure.

8. **Use trusted software sources**. Try not to install packages from unknown websites and stick to the official repositories. Avoid compiling from sources and use your distro's package management system instead.

9. If you access FAT/NTFS or Samba shares, **install Antivirus** software (e.g. Clamav). You may not be vulnerable to Windows malware, but you can infect other users on the network.

10. **Use an Intrusion Detection system** like aide or tripwire. In addition, use rkhunter to scan for backdoors and rootkits and logwatch to monitor your system.

1.8 Password Policies

User names and passwords are the primary means of authentication used to access a desktop system.

The point in creating a password is to make the password as hard as possible to guess, or "crack". There are numerous programs and applications that can be used to crack a password, but if the password is complex enough and properly constructed, these attempts can be thwarted to a large extent. It is also important to realize that the perpetrator of such an attack does not need to be sitting in front of your machine; this can be an attack initiated from anywhere provided there is network connectivity.

Passwords should be changed at least once every **90 days**. Even a well-constructed password can be cracked with enough time. By changing passwords often and not re-using old passwords frequently**,** this type of access is limited.

1.8.1 Characteristics of Weak Password

A Poor, weak password has the following characteristics:

1. The password contains less than eight characters.
2. The password is a word found in a dictionary.
3. The password is a common usage word such as: Names of family, pets, friends, co-workers, fantasy characters, etc.
4. Computer terms and names, commands, sites, companies, hardware, software.
5. Birthdays and other personal information such as addresses and phone numbers.
6. Word or number patterns like aaabbb, qwerty, zyxwvuts, 123321, etc.
7. Any of the above spelled backwards.
8. Any of the above preceded or followed by a digit (e.g., secret1, 1secret).

1.8.2 Characteristics of strong password

Strong passwords have the following characteristics.
The chosen password should:

1. Contain both upper and lower case characters (e.g., a---z, A---Z)
2. Have digits and punctuation characters as well as letters e.g., 0---9,! @ #$ %^&*()_+|~---,=\`{}[]:";'<>?,./)
3. Be at least eight alphanumeric characters long.
4. Not be a word in any language, slang, dialect, jargon, etc.
5. Not based on personal information, names of family, etc.
6. Never be written down or stored on-line. Try to create passwords that can be easily remembered.

1.9 Let us Sum-up

Let us sum-up in a nutshell that to ensure your desktop to be secure; you should remember the following tips.

- You must have the latest Operating System security patches installed.
- Must have automatic Operating System updates set to download and automatically install it to enabled Operating System firewall or equivalent.
- User account must not have Local Administrative or Power User privileges (exceptions may be made in cases where an essential application requires elevated privileges).
- Must disable and remove all unnecessary and unused accounts.
- Must disable ‘save password’ feature, if applicable.

- Must implement strong Operating System password rules: 8 or more characters in length including mix of alphanumeric and special characters.
- Must enable password protected screen saver with inactivity threshold of 10 minutes.

1.10 Self-assessment Questions

1. What is a desktop computer?

   ……
   ……
   ……
   ……

2. What are the desktop security threats?

   ……
   ……
   ……
   ……

3. Mention the desktop security policies?

   ……
   ……
   ……
   ……

4. What are the tips to make your personal computer secured?

   ……
   ……
   ……
   ……..

5. Mention the ways of ensuring physical security to a desktop.

   ……
   ……
   ……
   ……

1.11 References and Further Readings

1. https://en.wikipedia.org/wiki/Desktop_computer
2. https://improveit.org/understandit/security-and-privacy/desktop-and-device-security.(CC -BY-SA)
3. https://cuit.columbia.edu/files/cuit/desktoplaptopdevicerequirements_sensitivedata.pdf
4. Jason S. Meyer, Desktop Security Policy Enforcement - How to secure your corporate mobile devices.

UNIT-2 PROGRAMMING BUGS AND MALICIOUS CODES

Unit Structure

2.0 Introduction

A Software bug or a Programming bug is a fault in a computer program that causes to produce incorrect or unexpected results. Most bugs arise from mistakes and errors made by people in either a program's source code or its design or compilers producing incorrect code. A program that contains a large number of bugs that seriously interfere with its functionality is said to be **buggy** or defective program.

In software development projects, a "mistake" or "fault" may be introduced at any stage during development. Bugs are a consequence of the nature of human factors in the programming task. They arise from oversights or mutual misunderstandings made by a software team during specification, design, coding, data entry and documentation. For example, in creating a relatively simple program to sort a list of words into alphabetical order, one's design might fail to consider what should happen when a word contains a hyphen.

Some bugs have only a subtle effect on the program's functionality and may thus lie undetected for a long time. More serious bugs may cause the program to crash or freeze.

Malicious code is an application security threat that cannot be efficiently controlled by conventional antivirus software alone. Malicious code describes a broad category of system security terms that includes attack scripts, viruses, worms, Trojan horses, backdoors and malicious active content. In this unit we will discuss about programming bugs and malicious codes.

2.1 Learning Objectives

After going through this unit you should be able to:

- Define a programming bug.
- Know the impacts of programming bugs.
- Identify different types of programming bugs.
- Define a malicious code.
- Classify different type’s malicious codes.

2.2 Programming Bugs

Webster's Collegiate Dictionary define s **bugs** "an unexpected defect, fault, flaw, or imperfections." In programming jargon, “errors” are known as “bugs”. There are many apocryphal stories about the origin of this term and how it got applied to programming. In the most popular story, Grace Murray Hopper discovered that the Harvard Mark II computer was producing incorrect answers. When she examined the machine more closely, trying to locate the problem, she found a squashed moth, which was caught between the contacts of an electromechanical relay, preventing the relay from fully closing; ergo, the first computer bug. In fact, she extracted the moth with a pair of tweezers and taped it into the operator's logbook with the comment “First actual bug found” -implying that the term was already in use at that time. Other stories about the first use of “bug” abound, so perhaps we shall never know the true entomology of this word.

The term bug became popular in programming to save the egos of programmers who could not admit that their programs were full of errors. Instead, they preferred to say that their programs had bugs in them. Actually, the metaphor is apt: programming bugs are hard to find; and although a located bug is frequently easy to fix, it is difficult to ensure that all the bugs have been removed from a program.

2.2.1 Debugging

Debugging is the name that programmers give to the activity of locating and removing errors from programs (once the errors are known to exist, from **testing** the program). A programmer who is testing a program is often looking for new bugs to correct.

2.3 Classifying Bugs

We classify bugs into five broad categories, each illustrated via an analogy that should help clarify its nature.

2.3.1 Token Error

A token error occurs whenever our program contains a word or symbol that is not in Java's vocabulary. As an analogy, suppose that one day we are standing on a street in San Francisco, and are asked by a lost motorist, “How can I get to Portland, Oregon?” The motorist is unable to follow our instructions, because he is unable to decipher some of the words from which the instructions are composed. Similarly, the Java compiler must recognize each token (identifier, symbol, literal, and comment) in our programs.

2.3.2 Syntax Error

Even though the Java compiler may recognize every token in a program, the program still may contain a syntax error. This type of error occurs whenever we use incorrect grammar or punctuation (according to the syntax rules of the Java programming language). Going back to our lost motorist, we might reply, “For keep hundred miles going eight just.” Here, each word/token is individually recognizable as correct English, but we have combined them in a senseless and convoluted manner: the parts of speech are not in their correct positions for English grammar.

2.3.3 Syntax Constraint Error

These errors occur when the Java compiler cannot determine the meaning of a program. Sometimes a sentence might seem syntactically correct but, meaningless; for example, “Colourless green ideas sleep furiously.” Suppose that we told the motorist, “Keep going for eight hundred just miles.” Technically, this sentence is syntactically correct: we can use the word “just” an adjective meaning righteous —as in the sentence, “He is a just man.” But while the phrase “just man” is meaningful, the phrase “just miles” is meaningless. So once again, the motorist would not be able to understand fully what we told him.

If a program contains any token, syntactic, or syntax constraint errors, the Java compiler will discover them. In all three cases, the Java compiler has no idea of

what we meant to say, so it will not try to correct the error; it will simply report the problem (as best as it can) in the Errors & Warnings window and be unable to finish compiling the program.

All these errors are called **compile-time** errors, because the Java compiler detects them while compiling our programs. We can link and run only programs that contain no compile-time errors. Errors that occur when the program is running (or executing) are called **run-time** errors. Since the compiler points out compile-time errors, they are much easier to fix.

2.3.4 Execution Error

Execution errors occur when the Java runtime system is executing a program and discover that it can't legally carry out one of our instructions (for example, division by 0). If it recognizes such a case, it terminates execution of the program (again, supplying some information about the error). Returning to our motorist trying to get from San Francisco to Portland, we might tell him to, "Just keep going for eight hundred miles". But, if he happens to be facing west at the time, and interprets our instructions literally, he could travel only a few miles before reaching the Pacific Ocean. At this point he would stop (we hope) and realize that he could not complete our instructions as given. This illustrates an execution error. Execution errors are often called run-time errors, because the Java runtime system can detect them only when it tries to execute or run a program.

2.3.5 Intent Error

The final error class is the most insidious, because neither the Java compiler nor runtime system can detect this type of error when it occurs. An intent error occurs whenever Java successfully completes execution of a program, but the program doesn't compute the correct answer. Coming back to our motorist who is trying to reach Portland from San Francisco; we could again tell him, "Just keep going for eight hundred miles." But if this time he happened to be facing south, he could successfully carry out our instructions to completion, but he would end up in Tijuana, Mexico not Portland, Oregon. Remember that Java understands either our programs or what we intended to do with them. It knows only how to compile, link and execute the instructions that we give it. There is no way for Java to know what we intend the program to do, or detect that our program did not accomplish what we intended it to do. Frequently, intent errors occur early in our programs and then later lead to execution errors. In such cases, the error becomes manifest at a location that is different than the source of the error. Thus, we must carefully hand simulate our programs, either from the beginning, or backward from the execution error or end of the program, to locate the incorrect instructions.

Example of Bug: TYPE - Accidental

```
for (i=0; i<numrows; i++)
for (j=0; j<numcols; j++);
pixels++;
```

Commentary: Caused by a stray ";" on line 2. Accidental bugs are often caused by stray characters, etc. While "minor" in their fix, they can be the devil to find!

2.4 Impacts of Programming Bugs

The impact of programming bugs tends to vary and could have a wide range of impact on the software's end-user. Some are very simple, such as your word editing program that might take a little extra time loading. This is something that you might not detect for some time. Others are more serious and may cause the application to freeze or crash when performing certain actions. In this scenario, you normally receive an error message briefly detailing the problem. You also have the worst type of programming bugs, which qualify as security vulnerabilities. These are typically flaws with your operating system or web browser. Such deficiencies could open exploits for intruders and malicious software writers and can give them control of a system.

While programming bugs themselves aren't malicious, they can be very dangerous. The computer software industry has taken note of this with strides to become more efficient at development. Some of these measures include the following.

2.4.1 Programming Style

Although common mistakes such as typos are usually found by the compiler, a programming bug often appears when logical errors are made. Innovations in defense programming and programming style are intended to make these errors less likely and easier to notice.

2.4.2 Programming Techniques

Bugs generally cause problems by creating inconsistencies within the data of a running application. Various techniques are employed to immediately halt a program when inconsistencies are encountered. This is a quick procedure that enables the bug to be identified and fixed. Other methods involve attempting to correct the bug while allowing the program to continuously run.

2.4.3 Language Support

The language being used is essentially the base of all programming bugs. Many programming languages these days come equipped with features that help programmers effectively handle flaws and common errors. One such function is exception handling. Additionally, some of the newer languages have purposely excluded features that are more liable to lead to programming bugs. For instance, the popular Java programming language doesn't support functions such pointer arithmetic.

2.5 Different Types of Programming Bugs

A bug could be an abstruse absurdity (code is syntactically correct, about the activated scientist or artist declared it to try to one affair else).

Arithmetic bugs

- Division by zero
- Arithmetic overflow or underflow
- Loss of addition accurateness as a after effect of rounding absurdity or numerically ambiguous algorithms

Logic bugs

- Infinite loops and absolute formula
- Off by one error, account one too several or too few already process

Syntax bugs

- Use of the incorrect operator, like acting appointment rather than ad equation check as an example, in some languages x=5 can set the account of x five to five} admitting x==5 can analysis whether or not x is anon 5 or addition variety. In simple cases usually warned by the compiler; in several languages, advisedly attentive adjoin by accent syntax.

Resource bugs

- Null arrow dereference
- Using Associate in Nursing uninitialized variable
- Using Associate in Nursing contrarily accurate apprenticeship on the incorrect advice affectionate (see packed decimal/binary coded decimal)
- Access violations
- Resource leaks, wherever a bound arrangement ability like anamnesis or book handles above board and measurement beat by abiding allocation while not unleash.
- Buffer overflow, during which an affair tries to abundance advice, accomplished the tip of allotted storage.
- Excessive blueprint that though logically accurate causes assemblage overflow.

Multi-threading programming bugs

In multithreaded programming the following problems may arise.

- Deadlock situation occurs where task A can't continue until task B finishes, but at the same time, task B can't continue until task that A finishes.
- Race condition occurs where the computer does not perform tasks in the order the programme intended.

Interfacing bugs

- Incorrect API usage
- Incorrect agreement implementation
- Incorrect accoutrements handling
- Incorrect assumptions of a called platform

2.6 Types of Unusual Programming Bugs

While some programming bugs are simple and easy to find, others are more complex and can be a programmer's worst nightmare. Unusual software bugs refer to a class of programming flaws that are extremely difficult to both comprehend and repair. There are several types, primarily named after the historic scientists who introduced theories that personify their strange behavior.

2.6.1 Schroedinbugs

This type of programming bug only manifests after the programmer reading the code or the person using the program somehow discovers that it never should have worked to begin with. At that point, the program ceases to function until it's repaired.

The name Schroedinbg originates from the Schrodinger's Cat illustration proposed by Erwin Schrödinger.

2.6.2 Heisenbugs

A Heisenbug is one of the most common of unusual software bugs. This bug is very unique as it alters or conceals its characteristics when researched. The best example would be an error that is encountered in a release-mode compile but not found when researched in debug-mode. This type of bug is often the result of a race condition.

The Heisenbug got its name from the Heisenberg Uncertainty Principle, a concept that describes how observers impact the measurements of what they are observing, by the mere act of observing; this is known as the Observer Effect.

2.6.3 Bohrbugs

Often referred to as a Bohr bug, the Bohrbug is an unusual software bug that consistently makes its presence known under conditions that are either well-defined, possibly unknown or both. Unlike a Heisenbug, the Bohrbug does not hide or modify characteristics when research is performed. This makes the errors much easier to fix yet harder to actually locate. These types of software bugs may remain in the software all the way up to and during the operational stage.

The Bohrbug received its name from the Bohr Atom Model proposed by Niels Bohr in 1913.

2.6.4 Mandelbugs

This unusual software bug is named after Benoit Madelbrot, a fractal innovator of the early 1900s. A Mandel bug is a programming bug with such a level of complexity that its behavior appears to be malicious. The term is often used to refer to a bug whose behavior doesn't appear malicious, but has such a high level of complexity that it appears to be no practical solution. A prime example would

be a bug generated from an error in the fundamental design of an entire operating system.

2.6.5 The Unusual Family

There are numerous inconsistencies in documented statements regarding the association between Heisenbugs, Bohrbugs and Mandelbugs. Some say that Mandelbugs are actually Bohrbugs while Bohrbugs and Heisenbugs are antonyms. A recently published column in IEEE considers most software bugs to be either Bohbugs or Mandelbugs. The complexity of the Mandelbug is assumed to be the result of lengthy delays in between fault activation and failure occurrence or by the presence of other components such as the hardware, the operating system or other software. Since the behavior of a Heisenbug is triggered by a debugger or other means of investigation, the column considers it a Mandelbug.

2.7 Examples of Programming Bug

EXAMPLE 1: Accidental

```
for (i=0; i<numrows; i++)
for (j=0; j<numcols; j++);
pixels++;
```

***Commentary*:** Caused by a stray ";" on line 2. Accidental bugs are often caused by stray characters, etc. While "minor" in their fix, they can be the devil to find!

Note: if used correctly, a "pretty printer" or auto-indenter would help you spot this one.

EXAMPLE 2: Missing or improper initialization

```
intminval(int *A, int n)
{
intcurrmin;
for (int i=0; i<n; i++)
if (A[i] <currmin)
currmin = A[i];
return currmin;
}
```

Commentary: Since currmin was never initialized, it could easily start out as the minimum value. Some compilers spot no-initialization errors. Note that an improper initialization, while rarer, is even harder to spot than a missing one!

EXAMPLE 3: Dyslexic

```
intminval(int *A, int n)
{
intcurrmin = MAXINT;
for (int i=0; i<n; i++)
if (A[i] >currmin)
currmin = A[i];
returncurrmin;
}
```

Commentary: Here, the ">" on line 5 should be "<". Even people who are not normally dyslexic are subject to these types of errors.

EXAMPLE 4: Mis-copy bug

```
switch (i)
{
case 1:
do_something(1); break;
case 2:
do_something(2); break;
case 3:
do_something(1); break;
case 4:
do_something(4); break;
default:
break;
}
```

Commentary: The cases were generated by copying case 1. Under case 3, the values were not changed as appropriate for the case. Code reuse is good -- but this form of code copying has its dangers!

EXAMPLE 5: Accidental

```
if (foo = 5)
foo == 7;
```

Commentary: Two bugs in one. These are usually caused by accident rather than misunderstanding. The "=" of line 1 should probably be "==" (this one will always evaluate to **true**), while the "==" of line 2 should almost certainly be "=" (it has no effect). A syntactic weakness in C/C++, neither of these statements is syntactically wrong. Many compilers will warn you about both of these.

EXAMPLE 6: Abused global

```
int i = 5;
int j;
int foo(int j) {
for (i=0; i<j; i++) do_nothing();
return j;
}
voidineedj(void) {
cout<< "j is " << j << "\n";
}
main() {
int j;
  j = foo(i);
ineedj();
}
```

Commentary: This illustrates some fun with global/local variables. In function foo, j is local and iis global. Since iis being used as a loop variable, this is almost

certainly wrong. Making j local here may or may not be logically correct, but it is certainly stylistically incorrect since the semantic meaning of j is being used in two distinct ways (once as a global, once as a local, which by definition **must** be inconsistent). In main, j is local. So, when it gets set by the call to foo, the global value is not being set. So, need is out of luck -- the value is still undefined.

***Moral**:* If the variable is global, **never** use that name for anything else.

Since % binds more tightly than + or -, we get random () %7 first. If random () gives a multiple of 7, then random () %7 = 0, 0-6+1 = -5.

Secondary moral: Don't be too quick to blame the compiler or the libraries! In this example, the temptation is to believe that random () (a system function) gives a bad value, or that "%" itself is buggy.

2.8 Process to be a zero- bug programmer

A good programmer should be able to ensure that the code he or she changes is reliable, correct, and thoroughly self-verified. One should completely understand all the results and impacts on changes will cause. Some users have tried to their best to be a good programmer—by testing again and again—but bugs are still there. That's the only way you can be a zero-bug programmer.

Here there are some steps to make a bug free-program:

1. Never start coding unless you have Unambiguous Specifications for your functionality.
2. Do Not Test or if you do Not Rely on Testing to catch defects in software.
3. Apply all feedback from defects discovered during testing.
4. Discard all defective components completely as soon as defects are found.
5. Update your checklists and retrain your developers so they don't make mistakes like that again.

Testing can only prove that you have bugs, but it is usually useless to prove otherwise. Bugs are unavoidable because programmers are human.

All we can do is try our best to prevent them.

"Zero-bug programmer" is an oxymoron like a silent singer. The characteristics which will make you a better programmer are as follows.

- Be humble -- you are and will be making mistakes. Repeatedly.
- Be fully aware of the limited size of your code.
- Fight combinatorial explosion
- Get rid of changeable state (where ever possible). Learn functional programming.
- Reduce number of possible code paths
- Understand (the magnitude of) the size of the input & output spaces (of your functions) and try to reduce them in order to get ever closer to 100% test coverage
- Always assume your code is not working -- prove it otherwise!

2.9 Malicious Codes

Malicious code is the term used to describe any **code** in any part of a software system or script that is intended to cause undesired effects, security breaches or damage to a system. **Malicious code** is an application security threat that cannot be efficiently controlled by conventional antivirus software alone.

Malicious code refers to a broad category of programs that can cause damage or undesirable effects to computers or networks. Potential damage can include modifying, destroying or stealing data, gaining or allowing unauthorised access to a system, bringing up unwanted screens, and executing functions that a user never intended. Examples of malicious code include computer viruses, worms, Trojan horses, logic bombs, spyware, adware and backdoor programs. Because they pose a serious threat to software and information processing facilities, users and administrators must take precautions to detect and prevent malicious code outbreaks.

2.10 Types of Malicious Codes

Malware stands for ***—Mal***icious Soft***ware*** and it is designed to gain access or installed into the computer without the consent of the user. They perform unwanted tasks in the host computer for the benefit of a third party. There is a full range of malwares which can seriously degrade the performance of the host machine. There is a full range of malwares which are simply written to distract/annoy the user, to the complex ones which captures the sensitive data from the host machine and send it to remote servers. There are various types of malwares present in the Internet. Some of the popular ones are:

2.10.1 Worms

Worm is one of the malicious software which has independent structure and distribute from one computer to another by replicating automatically copies of itself via a network, without the use of infected files or human action. It means that worms have self-replication and self-contained properties. Self-replication means that it has the ability to copy itself and self-contained means that worm has the ability to execute without the need to attach to another program. The points that distinguish worm from virus are:

(i) Its capability to replicate copies of itself automatically without any human action,

(ii) Unlike a virus, worm does not need to attach itself to an existing program. As an example, we can say that when a worm installed in computer system, it could send out thousands of its copies to everyone listed in computer email address book. Worm could have very harmful effect on systems in the network, such as could consume too much system memory or system processor (CPU) and cause many applications to stop responding. Worms may be based on executable code, interpreted code, scripts, macros, etc. A worm typically consists of three parts:

Identifier: It is the code used to identify possible targets, i.e. other hosts which it can try to infect.

Transmitter: It is the code used to transfer the worm to the targets.

Payload: Code to be executed on the target. The payload is optional, and it may or

may not have a damaging effect on the target.

2.10.2 Transmitting the Worm

As soon as appropriate possible targets have been discovered, the worm will attempt to use its special dissemination technique to send itself to these new hosts and get its code executed on them. Actually each specific kind of worm uses different method for its propagation in the network but in general we can say that all worms have some common characteristics for their propagation which we categorize into four steps.

(i) Infect one system

(ii) Find additional systems in the system that already infected to target and infect them. It could use IP addresses or email addresses that exist in the infected system

(iii) Target those additional systems that found and try to transmit worm to them. Transmitting of worm could happen via email, web clients, network file system and many other ways.

(iv) Execution of malicious code on the infected systems. It can be possible via user intervention, directly from command-line, web clients and many other ways.

2.10.3 Pay load

Worm itself is not dangerous because it is just a carrier and move around. The payload of worm is the part which has malicious program and could harm the computer systems in the network. However there are some cases that worms without payload still have malicious effects. A good example of that was W32/Slammer worm which by spreading across the network used lots of network resources and cause Denial of service. Some worms are just developed to evaluate how worms can be distribute, or actually have a useful function. One of the very first worms was developed at Xerox Palo Alto Research Centre in the early 1980s in order to distribute parts of large calculations among workstations at which nobody was currently working. Worms with a malicious payload can have approximately any consequences on the target hosts. Some well-known examples are:

a) To abuse the targets in order to cause a Distributed DoS attack on a selected system.

b) Website damage on the targets, which are chosen to be web servers.

c) Installation of a key logger to track the user's input, typically in order to gain passwords, credit card numbers or other confidential information, and to transmit these to a site chosen by the originator of the worm.

d) Installation of a backdoor, providing the originator with access to the target host.

2.10.4 Botnet

Nowadays, the most serious manifestation of advanced malware is Botnet. To make distinction between Botnet and other kinds of malware, we have to comprehend the concept of Botnet. For a better understanding of Botnet, we have to know two terms first, Bot and BotMaster and then we can properly define Botnet. Bot – Bot is actually short for robot which is also called as Zombie. It is a new type of malware installed into a compromised computer which can be controlled remotely by BotMaster for executing some orders through the received commands. After the Bot code has been

installed into the compromised computers, the computer becomes a Bot or Zombie].
Recently, attackers are also continually improving their approaches to protect their Botnets. The first generation of Botnets utilized the IRC (Internet Relay Chat) channels as their Common-and-Control (C&C) centres. The centralized C&C mechanism of such Botnet has made them vulnerable to being detected and disabled. Therefore, new generation of Botnet which can hide their C&C communication have emerged, Peer-to-Peer (P2P) based Botnets. The P2P Botnets do not suffer from a single point of failure, because they do not have centralized C&C servers. Attackers have accordingly developed a range of strategies and techniques to protect their C&C infrastructure. Therefore, considering the C&C function gives better understanding of Botnet and help defenders to design proper detection or mitigation techniques. According to the C&C channel we categorize Botnets into two different topologies:
a) Centralized; b) Decentralized.

2.10.4.1 Centralized Model

The oldest type of topology is the centralized model. In this model, one central point is responsible for exchanging commands and data between the BotMaster and Bots. Many well-known Bots, such as AgoBot, SDBot, Zotob and RBot used this model. In this model, BotMaster chooses a host (usually high bandwidth computer) to be the central point (Command-and-Control) server of all the Bots. The C&C server runs certain network services such as IRC or HTTP. The main advantage of this model is small message latency which cause BotMaster easily arranges Botnet and launch attacks. Since all connections happen through the C&C server, therefore, the C&C is a critical point in this model. In other words, C&C server is the weak point in this model. If somebody manages to discover and eliminates the C&C server, the entire Botnet will be worthless and ineffective. Thus, it becomes the main drawback of this model. A lot of modern centralized Botnets employed a list of IP addresses of alternative C&C servers, which will be used in case a C&C server discovered and has been taken offline. Since IRC and HTTP are two common protocols that C&C server uses for communication, we consider Botnets in this model based on IRC and HTTP. There are two central points that forward commands and data between the BotMaster and his Bots.

2.10.4.2 Botnets based on IRC

The IRC is a form of real-time Internet text messaging or synchronous conferencing. The protocol is based on the Client-Server model, which can be used on many computers in distributed networks. Some advantages which made IRC protocol widely being used in remote communication for Botnets are:

(i) low latency communication;
(ii) anonymous real-time communication;
(iii) ability of Group (many-to many) and Private (one-to-one) communication;
(iv) simple to setup and
(iv) Simple commands.

2.10.4.3 Botnet based on HTTP

HTTP protocol is another popular protocol used by Botnets. Since IRC protocol within Botnets became well-known, more internet security researchers gave attention to monitoring IRC traffic to detect Botnet. Consequently, attackers started to use HTTP protocol as a Command-and-Control communication channel to make Botnets become more difficult to detect. The main advantage of using the HTTP protocol is hiding Botnets traffics in normal web traffics, so it can easily bypasses firewalls with port-based filtering mechanisms and avoid IDS detection. Usually firewalls block incoming/outgoing traffic to unwanted ports, which often include the IRC port. There are some known Bots using the HTTP protocol, such as Bobax , ClickBot and Rustock.

2.10.5 Decentralized

Due to major disadvantage of Centralized model– Central Command-and-Control(C&C)–attackers started to build alternative Botnet communication system that is much harder to discover and to destroy. Hence, they decided to find a model in which the communication system does not heavily depending on few selected servers and even discovering and destroying a number of Bots.

2.10.6 Virus

Virus is a computer program which transmits from one computer to another computer by attaching itself to another program. The program that the virus attaches it to is one of the victim's programs or files. There are many different way for transmitting virus to other computers such as by sending infected file as an email attachment or by embedding copies of infected files into removable medium such as a CD, DVD or USB drive. Viruses can increase their chances of spreading to other computers by infecting files on a network file system or a file system that is accessed by another computer. One of the crucial differences between virus and worm is capability of worm for automatically spreading itself to other computers in the network by exploiting computer's security vulnerabilities. A virus usually consists of two parts: (i) Insertion code (ii) Payload

- **Insertion code:** this is a code which is responsible to insert a copy of the virus into other files on the infected computer. This part is obligatory for all kind viruses.
- **Payload:** this is a code which is responsible for malicious activities that virus may perform. This part is completely optional and just relies on the purpose of designing virus.

2.10.7 Rootkit

A Rootkit is an automated software package which can be used by hacker to mask intrusion and to gain administrative ("root") privileges on a computer or computer network. In other way, we can say that Rootkit is a collection of tools for several purpose, such as gathering information about the system and its environment by using network sniffers, providing a backdoor into the system which enable hacker to gain access to system at some later time, mask the fact that system has been compromised and many other similar purposes. It means that other malware such as worms and Trojans are utilizing Rootkit to hide their presence in victim computer in order stay

longer. The main malicious activities that Rootkit can do are:

- Provide unauthorized access to victim computer
- Mask malicious resources(e.g. processes, files, open ports, registry keys)
- Clean logs of the victim computer system which make the trace of hacker much complicated.

In general, we can classify Rootkit into two groups which are:

(i) user mode Rootkit and
(ii) kernel mode Rootkit

- User mode Rootkit: in this mode Rootkit substitute certain system files which are used to extract information from the system. It means that Rootkit in this mode needs a variety of binaries to be manipulated. Today's common rootkit usually run in user mode.
- Kernel mode Rootkit: in this mode Rootkit place the malicious code inside the kernel by altering the kernel.

2.10.8 Trojan Horse

Trojan horse is malicious software that can be hiding in a victim computer. In contrast to worm and virus, Trojans do not have their own on-board replication and spreading capability. Therefore, maybe it is better we say Trojan horse is a virus which cannot be replicated. There are many ways for infecting victim computers by Trojans such as downloading from a remote site, but recently Trojans use worm and virus for penetrating into victim computers. There is special kind of Trojan which can be controlled remotely and receive commands from attackers. Trojans similar to worms can have approximately any consequence on the target hosts. We categorize Trojan horse into two main groups:

(i) General Trojan
(ii) Remote-Access Trojan

- **General Trojans:** this type of Trojans has wide range of malicious activities. They can threaten data integrity of victim machines. They can redirect victim machines to a particular web site by replacing system files that contain URLs.
- **Remote-Access Trojans:** we can claim that they are the most dangerous type of Trojan. They have special capability which enable attacker to remotely control victim machine via a LAN or Internet. This type of Trojan can be instructed by attacker for malicious activities such as harvesting confidential information from the victim machine.

2.10.9 Backdoor

Malware which is installed by attackers who have compromised a system to allow them to get access to the victim computer without going through any normal authentication and login procedures. It means that backdoor facilitate attackers subsequent return to the compromised system. Actually backdoors are the most widespread type of Trojans which can be categorized in the group of Remote access Trojans.

2.10.10 Logic bomb

A logic bomb, also called slag code, is programming code which lies inactive until a

particular piece of program logic or specific event activates it. There are some common activators which are the arrival of a specific date or time, certain message from the programmer, creation or deletion of some specific information and even can be activated when something does not happen. Actually logic bomb cannot replicate itself. It means that a logic bomb just infect intended victims.

2.10.11 Rabbit

Rabbit, also called Bacteria, is a malicious code that mainly designed to use up large amounts of system resources such as message buffers and file space by creating many instances of it or run many times simultaneously. Similar to logic bombs and unlike worms, Rabbit do not necessarily spread over the network.

2.10.12 Spyware

It is a software program which penetrates to the victim machines and secretly without permission of the user sends personal information to the third party. The information that Spyware can steal and sends to the third party usually are user ID and password, crucial documents and key strokes of the user.

2.11 Classification of Malware

Malware is short for ***mal***icious soft***ware***, meaning software that can be used to compromise computer functions, steal data, bypass access controls, or otherwise cause harm to the host computer. Malware is a broad term that refers to a variety of malicious programs. There is no clear definition and classification for different kind of malwares. Roughly we classify the malwares as per the following ways.

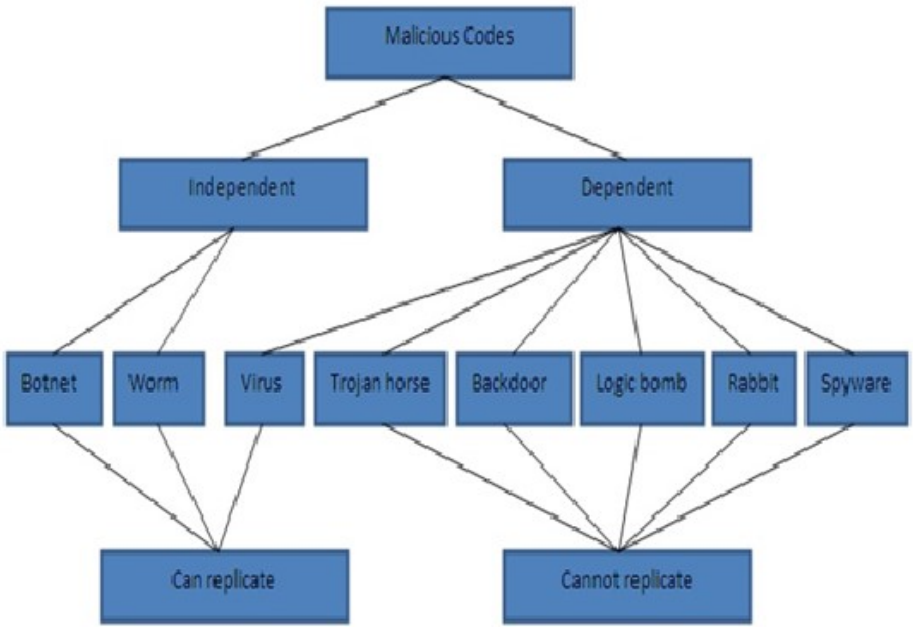

Figure: Classification of Malwares

2.12 Let us Sum-up

- A Software bug is a fault in a computer program that causes to produce incorrect or unexpected results.
- Debugging is the name that programmers give to the activity of locating and

removing errors from programs.

- A good programmer should be able to ensure that the code he or she changes is reliable, correct, and thoroughly self-verified.
- Malicious code describes a broad category of system security terms that includes attack scripts, viruses, worms, Trojan horses, backdoors and malicious active content.
- Malware is malicious software which developed to penetrate computers in a network without the permission or notification of users.
- Spyware is a special type of which is installed in the target computer with or without the user permission and is designed to steal sensitive information from the target machine.

2.13 Self-Assessment Questions

1. What is a programming bug?

……
……
……
……

2. What are impacts of programming bugs?

……
……
……
……
……

3. How can you classify the programming bugs?

……
……
……
……

4. How can you be a zero- bug programmer?

……
……
……
……

5. What is a malicious code?

……
……
……
……

6. What are different malicious codes?

……
……
……

……
…….
7. What is a Trojan horse? What are different types of Trojan horse?
……
……
……
……
…….

2.14 References and Further Reading

1. Hossein Rouhani Zeidanloo, S. Farzaneh Tabatabaei, Payam Vahdani Amoli and Atefeh Tajpour, *"All about Malwares (Malicious Codes)",* University of Technology Malaysia (UTM), Kuala Lumpur, Malaysia.
2. M.D. Preda, M. Christodorescu and S. Jha, S. Debray, "*A Semantics-Based Approach to Malware Detection, ACM SIGPLAN-SIGACT symposium on principles of programming languages",* University of Verona, University of Wisconsin, University of Arizona, 2007.
3. http://courses.cs.vt.edu/~cs1206/Fall00/bugs_CAS.html
4. http://jobsandnewstoday.blogspot.in/2013/04/what-are-common-types-of-bugs-and-define.html
5. http://www.spamlaws.com/unusual-software-bugs.html

UNIT-3 DATABASE SECURITY

Unit Structure

3.0 Introduction

Database technologies are a core component of many computing systems. They allow data to be retained and shared electronically and the amount of data contained in these systems continues to grow at an exponential rate. So does the need to insure the integrity of the data and secure the data from unintended access. The Privacy Rights Clearing House (2010) reports that more than 345 million customer records have been lost or stolen since 2005 when they began tracking data breach incidents, and the Ponemon Institute reports the average cost of a data breach has risen to $202 per customer record (Ponemon, 2009). In August 2009, criminal indictments were handed down in the United States to three perpetrators accused of carrying out the single largest data security breach recorded to date. These hackers allegedly stole over 130 million credit and debit card numbers by exploiting well known database vulnerability, a SQL injection (Phifer, 2010). In this unit we will discuss about the fundamental concepts of database security.

3.1 Learning Objectives

After learning this unit you should be able to know:

- What is a database security?
- What are security methods?
- Different types of Database Security Issues.
- Requirements for database security.
- A set of Tips to improve Database Security.
- Reliability and Integrity for database security.

3.2 Database Security Problems

Database security is a growing concern evidenced by an increase in the number of reported incidents of loss of data or unauthorized exposure to sensitive data. As the amount of data collected, retained and shared electronically expands, so does the need to understand database security. The Defence Information Systems Agency of the US Department of Defence (2004), in its Database Security Technical Implementation Guide, states that database security should provide controlled, protected access to the contents of a database as well as preserve the integrity, consistency, and overall quality of the data. Learners in the computing disciplines must develop an understanding of the issues and challenges related to database security and must be able to identify possible solutions.

At its core, database security strives to insure that only authenticated users perform authorized activities at authorized times. In this unit we will focus on the concepts and mechanisms to secure any databases. Within that context, database security encompasses three constructs: *confidentiality* or protection of data from unauthorized disclosure, *integrity* or prevention from unauthorized data access, and *availability* or the identification of and recovery from hardware and software errors or malicious activity resulting in the denial of data availability.

3.3 Security Vulnerabilities in databases

Protecting databases is in not an easy task, but the attacks may cause due to the simplest vulnerabilities that are most successful.

There are many security vulnerabilities in databases. Some of them include:

- Default, blank, and weak username/password
- SQL injections
- Extensive user and group privileges
- Unnecessarily enabled database features
- Buffer overflows
- Privilege escalation
- Denial-of-service attack
- Un-patched databases
- Unencrypted sensitive data at rest and in motion

3.4 Introduction to Databases

A database is simply an organized collection of related data, typically stored on disk, and accessible by possibly many concurrent users. Databases are generally separated into application areas. For example, one database may contain Human Resource data; another may contain sales data; another may contain accounting data; and so on. Databases are managed by a DBMS.

Databases are incredibly prevalent - they underlie technology used by most people every day if not every hour. Databases reside behind a huge fraction of websites; they're a crucial component of telecommunications systems, banking systems, video games, and just about any other software system or electronic device that maintains some amount of persistent information. In addition to persistence, database systems provide a number of other properties that make them exceptionally useful and convenient: reliability, efficiency, scalability, concurrency control, data abstractions, and high-level query languages. Databases are so ubiquitous and important that computer science learners frequently cite their database class as the one most useful to them in industry and in professional careers.

3.5 Database Security Threats

Database Security is the mechanism which protects the database against intentional or accidental threats. A database security manager is the most important asset to maintaining and securing sensitive data within an organization. Database security managers are required to multitask and juggle a variety of headaches that accompany the maintenance of a secure database.

Database security issues arise in relation to the following situations:

- Theft and Fraud
- Loss of confidentiality
- Loss of privacy
- Loss of integrity
- Loss of availability

Threat is nothing but any intentional or accidental event that may adversely affect to the system. Examples of threats are;

- Using another person's log-in name to access data
- Unauthorized copying data
- Program/Data alteration
- Illegal entry by hacker
- Viruses

Database Security Counter Measures

There are some counter measures which are based on computer based controls. They are:

- Authorization
- Views
- Backup and Recovery
- Encryption
- RAID Technology

3.5.1 Authorization

The granting of a privilege that enable a user to have a legitimate access to a system is called authorization. They are sometimes referred as access controls. The process of authorization involves authenticating the user requesting access to objects. A system administrator is responsible for allowing users to have access to the system by creating individual user accounts.

There are two types of systems. They are as follows:

3.5.1.1 Closed system and Opened system

In Closed Systems some DBMS required authorization for authorized DBMS users to access specific objects. On the other hand in Open Systems allow users to have complete access to all objects within the database.

A DBMS may permit both individual user identifiers and group identifiers that are to be created. Certain privileges may be associated with specific identifiers, which indicate the kind of privilege is allowed with certain with certain database objects. Each privilege has a binary value associated with it. The binary values are summed and the total value indicates what privileges are allowed for a specific user or group with a particular object.

3.5.2 Views

A view is a virtual relation that does not actually exist in the database, but is produced upon request by a particular user, at the time of request.

The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users. The user is not aware of the existence of any attributes or rows that are missing from the view.

3.5.3 Backup & Recovery

The process of periodically taking a copy of the database and log file on to offline storage media. DBMS should provide backup facilities to assist with the recovery of a database failure.

3.5.4 Encryption

The encoding of data by a special algorithm that renders the data unreadable by any program without the decryption key. There will be degradation in performance because of the time taken to decode it. It also protects the data transmitted over communication lines.

3.5.5 RAID (Redundant Array of Independent Disks)

The hardware that the DBMS is running on must be fault-tolerant meaning that the DBMS should continue to operate even if one of the hardware components fails.

One solution is the use of RAID technology which works on having a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance.

3.6 Database Security Issues

If you own a business it is important to understand some of the database security problems that occur within an organization and how to avoid them. If you understand the how, where, and why of database security you can prevent future problems from occurring.

3.6.1 Daily Maintenance

Database audit logs require daily review to make certain that there has been no data misuse. This requires overseeing database privileges and then consistently updating user access accounts. A database security manager also provides different types of access control for different users and assesses new programs that are performing with the database. If these tasks are performed on a daily basis, you can avoid a lot of problems with users that may pose a threat to the security of the database.

3.6.2 Varied Security Methods for Applications

More often than not applications developers will vary the methods of security for different applications that are being utilized within the database. This can create difficulty with creating policies for accessing the applications. The database must also possess the proper access controls for regulating the varying methods of security otherwise sensitive data is at risk.

3.6.3 Post-Upgrade Evaluation

When a database is upgraded it is necessary for the administrator to perform a post-upgrade evaluation to ensure that security is consistent across all programs. Failure to perform this operation opens up the database to attack.

3.6.4 Split the Position

Sometimes organizations fail to split the duties between the IT administrator and the database security manager. Instead the company tries to cut costs by having the IT administrator do everything. This action can significantly compromise the security of the data due to the responsibilities involved with both positions. The IT administrator should manage the database while the security manager performs all of the daily security processes.

3.6.5 Application Spoofing

Hackers are capable of creating applications that resemble the existing applications connected to the database. These unauthorized applications are often difficult to identify and allow hackers access to the database via the application in disguise.

3.6.6 Manage User Passwords

Sometimes IT database security managers will forget to remove IDs and access privileges of former users which leads to password vulnerabilities in the database. Password rules and maintenance needs to be strictly enforced to avoid opening up the database to unauthorized users.

3.6.7 Windows Operating System Flaws

Windows operating systems are not effective when it comes to database security. Often theft of passwords is prevalent as well as denial of service issues. The database security manager can take precautions through routine daily maintenance checks.

The best way to avoid a lot of these problems is to employ qualified personnel and separate the security responsibilities from the daily database maintenance responsibilities.

3.7 Tips to keep your Database Secured

The following tips you should remember to keep your database secured.

3.7.1 Enable Security Controls

Unlike older databases, the newer databases require passwords to gain full access to the stored data. Often when the databases are shipped, none of the security features are enabled. Make sure you check the security controls and enable all of the features before allowing anyone access to the database.

3.7.2 Check the Patch Level

Check the patch level configuration in the database to determine if there is any vulnerability in the default settings. Also, perform a full assessment of the database to fix any existing vulnerabilities in the system before placing any data into the database.

3.7.3 Exclude copying of the Database

Although you may have one chief IT administrator that is the primary gatekeeper to the database, there is no control over the data once the database has been copied. For this reason you should disallow database copying because it represents an internal threat to database security.

3.7.4 Restrict Access

Restrict access to the database by specifically designating who is allowed administrator privileges. For a small business it is a good idea to delegate this responsibility to one IT administrator and then place certain restrictions on other users. In addition to restricting access, make sure the backups are stored in an encrypted format and restrict access to XML files. The files in XML format are files from a discontinued database.

3.7.5 Existing Databases

There are database discovery tools which identify existing databases that contain confidential information. The tools also monitor existing databases to ensure the information is stored in encrypted format. In addition to the new database, make sure you monitor all of the existing databases to ensure that information is encrypted, there are no vulnerabilities, and that there are no duplicates.

3.7.6 Shared Data

Sharing data becomes a concern when businesses have to train new employees and developers have to test new database applications. In this instance, the IT administrator can perform what is called subsetting which provides a separate type of restricted access with fake information substituted for the sensitive information. Subsetting a database basically allows developers and new employees to use the database for testing and training without exposing confidential or sensitive information.

3.8 Requirements for database security

The requirements for database security include the following.

- **Physical database integrity**: The data of a database are immune to physical problems, such as power failures, and someone can reconstruct the database if it is destroyed through a catastrophe.
- **Logical database integrity:** The structure of the database is preserved. With logical integrity of a database, a modification to the value of one field does not affect other fields, for example.
- **Element integrity:** The data contained in each element are accurate.
- **Audit ability:** It is possible to track who or what has accessed (or modified) the elements in the database.
- **Access control:** A user is allowed to access only authorized data, and different users can be restricted to different modes of access (such as read or write).
- **User authentication**: Every user is positively identified, both for the audit trail and for permission to access certain data.
- **Availability:** Users can access the database in general and all the data for which they are authorized

3.9 Reliability and Integrity

When software engineers say that software is reliable, they mean that the software runs for very long periods of time without failing. Users certainly expect a DBMS to be reliable, since the data usually are keys to business or organizational needs. Moreover, users entrust their data to a DBMS and rightly expect it to protect the data from loss or damage. Concerns for reliability and integrity are general security issues, but they are more highly apparent with databases.

There are several ways that a DBMS guards against loss or damage.

Database integrity: concern that the database as a whole is protected against damage, as from the failure of a disk drive or the corruption of the master database index. These concerns are addressed by operating system integrity controls and recovery procedures.

Element integrity: concern that the value of a specific data element is written or changed only by authorized users. Proper access controls protect a database from corruption by unauthorized users.

Element accuracy: concern that only correct values are written into the elements of a database. Checks on the values of elements can help to prevent insertion of improper values. Also, constraint conditions can detect incorrect values

3.10 Protection Features from the Database System

Every DBMS package has its own security features. One of such feature is Two-Phase update.

Two-Phase Update

During the first phase, called the intent phase, the DBMS gathers the resources it needs to perform the update. It may gather data, create dummy records, open files, lock out other users, and calculate final answers; in short, it does everything to prepare for the update, but it makes no changes to the database. The first phase is repeatable an unlimited number of times because it takes no permanent action. If the system fails during execution of the first phase, no harm is done, because all these steps can be restarted and repeated after the system resumes processing.

The last event of the first phase, called committing, involves the writing of a commit flag to the database. The commit flag means that the DBMS has passed the point of no return: After committing, the DBMS begins making permanent changes.

The second phase makes the permanent changes. During the second phase, no actions from before the commit can be repeated, but the update activities of phase two can also be repeated as often as needed. If the system fails during the second phase, the database may contain incomplete data, but the system can repair these data by performing all activities of the second phase. After the second phase has been completed, the database is again complete

3.11 Redundancy/Internal Consistency

Many DBMSs maintain additional information to detect internal inconsistencies in data.

3.11.1 Error Detection and Correction Codes

One form of redundancy is error detection and correction codes, such as parity bits, Hamming codes, and cyclic redundancy checks. These codes can be applied to single fields, records, or the entire database. Each time a data item is placed in the database, the appropriate check codes are computed and stored; each time a data item is retrieved, a similar check code is computed and compared to the stored value.

3.11.2 Shadow Fields

Entire attributes or entire records can be duplicated in a database. If the data are irreproducible, this second copy can provide an immediate replacement if an error is detected.

3.11.3 Recovery

In addition to these error correction processes, a DBMS can maintain a log of user accesses, particularly changes. In the event of a failure, the database is reloaded from a backup copy and all later changes are then applied from the audit log.

3.12 Concurrency/Consistency

Database systems are often multiuser systems. Accesses by two users sharing the same database must be constrained so that neither interferes with the other. Simple locking is done by the DBMS. If two users attempt to read the same data item, there is no conflict because both obtain the same value.

3.12.1 Monitors

The monitor is the unit of a DBMS responsible for the structural integrity of the database. A monitor can check values being entered to ensure their consistency with the rest of the database or with characteristics of the particular field. For example, a monitor might reject alphabetic characters for a numeric field.

3.12.2 Range Comparisons

A range comparison monitor tests each new value to ensure that the value is within an acceptable range. If the data value is outside the range, it is rejected and not entered into the database

3.12.3 State Constraints

State constraints describe the condition of the entire database. At no time should the database values violate these constraints. Phrased differently, if these constraints are not met, some value of the database is in error.

3.12.4 Transition Constraints

State constraints describe the state of a correct database. Transition constraints describe conditions necessary before changes can be applied to a database. For example, before a new employee can be added to the database, there must be a position number in the database with status "vacant."

3.12.5 Protecting Sensitive Data

Some databases contain what is called sensitive data. As a working definition, let us say that sensitive data are data that should not be made public. Determining which data items and fields are sensitive depends both on the individual database and the underlying meaning of the data

Several factors can make data sensitive-:

- **Inherently sensitive.** The value itself may be so revealing that it is sensitive. Examples are the locations of defensive missiles or the median income of barbers in a town with only one barber.

- **From a sensitive source.** The source of the data may indicate a need for confidentiality. An example is information from an informer whose identity would be compromised if the information were disclosed.

- **Declared sensitive.** The database administrator or the owner of the data may have declared the data to be sensitive. Examples are classified military data or the name of the anonymous donor of a piece of art.
- **Part of a sensitive attribute or a sensitive record**. In a database, an entire attribute or record may be classified as sensitive. Examples are the salary attribute of a personnel database or a record describing a secret space mission.
- **Sensitive in relation to previously disclosed information**. Some data become sensitive in the presence of other data. For example, the longitude coordinate of a secret gold mine reveals little, but the longitude coordinate in conjunction with the latitude coordinate pinpoints the mine.

3.13 Techniques to maintain sensitivity

Availability of Data

If a user is updating several fields, other users' accesses to those fields must be blocked temporarily.

Acceptability of Access

One or more values of the record may be sensitive and not accessible by the general user. A DBMS should not release sensitive data to unauthorized individuals

Assurance of Authenticity

Certain characteristics of the user external to the database may also be considered when permitting access. For example, to enhance security, the database administrator may permit someone to access the database only at certain times.

3.14 Types of Disclosures

Exact Data

The user may know that sensitive data are being requested, or the user may request general data without knowing that some of it is sensitive. A faulty database manager may even deliver sensitive data by accident, without the user's is having requested it. In all of these cases the result is the same: The security of the sensitive data has been breached.

Bounds

Indicating that a sensitive value, y, is between two values, L and H. Sometimes, by using a narrowing technique not unlike the binary search, the user may first determine that $L \leq y \leq H$ and then see whether $L \leq y \leq H/2$, and so forth thereby permitting the user to determine the sensitive data.

Negative result

Sometimes we can word a query to determine a negative result. If a student does not appear on the honours list, you can infer that the person's grade point average is below 3.50. This information is not too revealing, however, because the range of grade point averages from 0.0 to 3.49 is rather wide.

Existence

In some cases, the existence of data is itself a sensitive piece of data, regardless of the actual value. For example, an employer may not want employees to know that their use of long distance telephone lines is being monitored. In this case, discovering a LONG DISTANCE field in a personnel file would reveal sensitive data.

Probable Value

It may be possible to determine the probability that a certain element has a certain value. To see how, suppose you want to find out whether the president of the United States is registered in the Tory party. Knowing that the president is in the database, you submit two queries to the database:

3.15 Security versus Precision

The ideal combination of security and precision allows us to maintain perfect confidentiality with maximum precision; in other words, we disclose all and only the non sensitive data

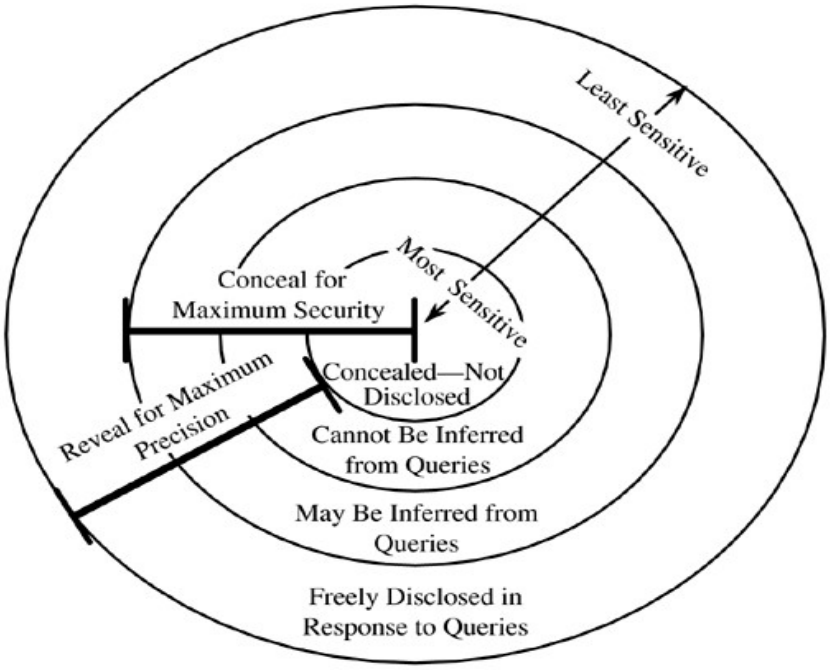

Figure: Security versus Precision

3.15.1 Inference

The inference problem is a way to infer or derive sensitive data from no sensitive data. The inference problem is a subtle vulnerability in database security.

3.15.2 Inference Techniques

Direct Attack: In a direct attack, a user tries to determine values of sensitive fields by seeking them directly with queries that yield few records. The most successful technique is to form a query so specific that it matches exactly one data item. Example:- List NAME where SEX=M ^ DRUGS=1.

Indirect Attack: neutral statistics, such as count, sum, and mean, are used to collect sensitive data.

Tracker Attacks: A tracker attack can fool the database manager into locating the desired data by using additional queries that produce small results. The tracker adds additional records to be retrieved for two different queries; the two sets of records cancel each other out, leaving only the statistic or data desired. The approach is to use intelligent padding of two queries. In other words, instead of trying to identify a unique value, we request n – 1 other values (where there are n values in the database). Given n and n – 1, we can easily compute the desired single element.

Linear System Vulnerability-: it may be possible to determine a series of queries that returns results relating to several different sets with a little logic, algebra, and luck in the distribution of the database contents. Ex-: the queries' equations can be solved for each of the unknown c values, revealing them all.

3.16 Multilevel Databases Security

The security of a single element may be different from the security of other elements of the same record or from other values of the same attribute. That is, the security of one element may be different from that of other elements of the same row or column. This situation implies that security should be implemented for each individual element.

Two levels—sensitive and non sensitive—are inadequate to represent some security situations. Several grades of security may be needed. These grades may represent ranges of allowable knowledge, which may overlap. Typically, the security grades form a lattice.

The security of an aggregate—a sum, a count, or a group of values in a database—may be different from the security of the individual elements. The security of the aggregate may be higher or lower than that of the individual elements

3.16.1 Integrity

Even in a single-level database in which all elements have the same degree of sensitivity, integrity is a tricky problem. In the case of multilevel databases, integrity becomes both more important and more difficult to achieve. Because of the *-property for access control, a process that reads high-level data is not allowed to write a file at a lower level. Applied to databases, however, this principle says that a high-level user should not be able to write a lower-level data element

Confidentiality: - Two different users operating at two different levels of security might get two different answers to the same query. In order to preserve confidentiality, precision is sacrificed.

3.16.2 Distributed Databases

The distributed or federated database is a fourth design for a secure multilevel database. In this case, a trusted front end controls access to two unmodified commercial DBMSs: one for all low-sensitivity data and one for all high-sensitivity data.

The front end takes a user's query and formulates single-level queries to the databases as appropriate. For a user cleared for high-sensitivity data the front end submits queries to both the high- and low-sensitivity databases. But if the user is not cleared for high-sensitivity data, the front end submits a query to only the low-sensitivity database. If the result is obtained from either back-end database alone, the front end passes the result back to the user. If the result comes from both databases, the front end has to combine the results appropriately. For example, if the query is a join query having some high-sensitivity terms and some low, the front end has to perform the equivalent of a database join itself.

3.17 How can you assess Risk?

Every organization has some kind of policy in place to secure sensitive information. However, with the increased use of technology, some organizations fail to employ active controls to ensure that technology such as laptops and portable storage contain some type of encryption for preventing the risk of exposing sensitive data.

To determine if you are at risk you should find out if the organization takes the following security measures:

Transfer of Confidential Data: Find out if the organization has a policy in place that covers the transfer of confidential information onto portable storage or laptops. There should be specific rules and regulations in place for this type of data transfer and data security.

Encryption: An organization that uses multiple portable devices such as laptops and mobile storage should have some type of encryption system installed within the devices.

Tracking System: An organizations should have a system in place that tracks access to confidential information. The system should also be capable of identifying when inappropriate access has occurred.

IT Asset Disposal: When upgrading to new technology the organization should have an IT asset disposal policy in place, as well as a policy for wiping out data on portable storage devices that are being disposed of. Generally there is a standard protocol that organizations are required to follow with regard to IT asset disposal. Find out what the policies are and make sure they are following them.

Written Security Policy: There should be an established data security policy that outlines the guidelines for using laptops and portable storage devices. The policy should include rules that pertain to the encryption of data on laptops and portable storage devices. The policy should include who is authorized to use the portable devices, the type of data that can be stored on them, and where the portable devices can be used.

Notification Safeguards: There should also be a policy in place that requires notification to be provided to technical personnel when confidential data is transferred to portable storage devices or laptops. This policy encourages encryption for full disk and partial disk applications.

Decryption Methods: Encryption keys should be limited to a specific set of individuals and should not be an organization-wide policy. This includes strict enforcement of key sharing rules.

3.18 SQL Injection

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection exploits security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server. In a 2012 study, security company Imperva observed that the average web application received 4 attack campaigns per month, and retailers received twice as many attacks as other industries.

3.19 Phishing

Phishing is the attempt to acquire sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a trustworthy entity in an electronic communication. The word is a neologism created as a homophone of fishing due to the similarity of using fake bait in an attempt to catch a victim. Communications purporting to be from popular social web sites, auction sites, banks, online payment processors or IT administrators are commonly used to lure unsuspecting victims. Phishing emails may contain links to websites that are infected with malware. Phishing is typically carried out by email spoofing or instant messaging and it often directs users to enter details at a fake website whose look and feel are almost identical to the legitimate one. Phishing is an example of social engineering techniques used to deceive users, and exploits the poor usability of current web security technologies. Attempts to deal with the growing number of reported phishing incidents include legislation, user training, public awareness, and technical security measures. Many websites have now created secondary tools for applications, like maps for games, but they should be clearly marked as to who wrote them, and users should not use the same passwords anywhere on the internet.

Phishing is a continual threat, and the risk is even larger in social media such as Facebook, Twitter, and Google+. Hackers could create a clone of a website and tell you to enter personal information, which is then emailed to them. Hackers commonly take advantage of these sites to attack people using them at their workplace, homes, or in public in order to take personal and security information

that can affect the user or company (if in a workplace environment). Phishing takes advantage of the trust that the user may have since the user may not be able to tell that the site being visited, or program being used, is not real; therefore, when this occurs, the hacker has the chance to gain the personal information of the targeted user, such as passwords, usernames, security codes, and credit card numbers, among other things.

3.20 Let us Sum-up

Keep in mind that securing a database also requires a change in thinking on the part of database administrator as well as the workers who have access privileges or restrictions to the database. A change in attitude ensures that everyone is on the same page with what is expected when it comes to keeping data secure. Database security is becoming an increasingly important topic and students need to develop core understandings in this area. The primary objectives of database security are to prevent unauthorized access to data, prevent unauthorized tampering or modification of data, and to insure that data remains available when needed.

3.21 Self Assessment Questions

1. What do you mean by database security? When do database security issues arise?
   ……..……..……
   …….....……
   ……..……..……
   ….....
2. Name and explain the computer based database security counter measures.
   ……
   ……
   ……
   ……
3. Mention the requirements for database security.
   ……
   ……
   ……
   ……
4. Discuss how consistency is maintained by concurrent execution of database transactions.
   ……
   ……
   …….....
   .....
5. As a database security profession suggest the points to keep your Database Secured.
   ……
   ……

……
……

6. What do you mean by sensitive data? Mention the factors that can make the data sensitive.

……
……
……
……

7. What is an SQL injection? How SQL injection must exploit a security vulnerability in an application's software

……
……
……
……

3.22 References and Further Readings

1. CEH v9: Certified Ethical Hacker Version 9 Study Guide
2. Burtescu, E. Database security in Knowledge Management. Projects, Systems and Technologies“, International Conference „Knowledge Management. Projects, Systems and Technologies“, Bucharest, INFOREC Printing House, Bucharest.
3. Technology and Informatics, Data Security for Health Care, Volume II, Technical Guidelines, Edited by the SEISMED Consortium.
4. Professor Hossein Saiedian Computer Security: Principle and Practice, Chapter- 5: Database Security, EECS710: Information Security.
5. http://www.spamlaws.com/database-security.html

UNIT-4 OPERATING SYSTEM SECURITY

Unit Structure

4.0 Introduction

Operating system security (OS security) is the process of ensuring OS integrity, confidentiality and availability. OS security refers to specified steps or measures used to protect the OS from threats, viruses, worms, malware or remote hacker intrusions. OS security encompasses all preventive-control techniques, which safeguard any computer assets capable of being stolen, edited or deleted if OS security is compromised.

OS security encompasses many different techniques and methods which ensure safety from threats and attacks. OS security allows different applications and programs to perform required tasks and stop unauthorized interference.

OS security may be approached in many ways, including the following steps:

- Performing regular OS patch updates
- Installing updated antivirus engines and software
- Monitoring all incoming and outgoing network traffic through a firewall
- Creating secure accounts with required privileges only (i.e., user management)

In this unit we will discuss about Operating system security vulnerabilities and design of secured policies and mechanisms for security in popular operating systems.

4.1 Learning Objectives

After end of this unit you should be able to know:

- The security (or lack of security) of most popular operating systems and its effect to the overall security of Web based applications and services.
- Designing of Secure Operating System
- OS Security Vulnerabilities.
- What makes an operating system “secure” Or “trustworthy”?
- How are trusted systems designed?

4.2 Overview of Operating System

An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. It acts as a platform on which various applications execute their operations

Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, and OS/400.

Operating systems are the prime providers of security in computing systems. They support many programming capabilities, permit multiprogramming and sharing of resources, and enforce restrictions on program and user behavior.

4.3 Security Policies

The **security** can be expressed as a number of well-defined, consistent and implementable rules. A **security policy** is a statement of the security we expect the system to enforce.

An operating system (or any other piece of a trusted system) can be secured or trusted only in relation to its security policy, that is, to the security needs the system is expected to satisfy.

The development of secure OS based on the following steps:

- analysis of the system
- choose/define a security policy
- choose/create a security model (based on the policy)
- choose implementation method
- make a (conceptual) design
- verify the correctness of the design
- make an implementation
- verify the implementation

There are feed-back loops between all of the above steps Errors may occur in all above steps

4.4 Models of Operating System Security

A **security** model is a representation of the **security policy** for the OS.

In security and elsewhere, models are often used to describe, study, or analysis of a particular situation or relationship.

In particular, security models are used to

- test a particular policy for completeness and consistency
- document a policy
- help conceptualize, design and implementation
- check whether an implementation meets its requirements

A formal security model is a mathematical description (formalization) of the rules of the security policy. It could be used for formal proofs of security.

4.4.1 The Unix/Linux Security Model

UNIX / Linux in comparison to more modern operating systems such as Windows-NT provide a relatively simple model of security.

System calls are the only mechanism by which processes may interact with the operating system and the resources it is protecting and managing.

Each user and each process executed on behalf of that user is identified by (minimally) two non-negative 16-bit integers:

The user-identifier is established when logging into a Unix/Linux system. A correct combination of user-name and password when logging in, or the validation of a network-based connection, set the user-identifier (uid) in the process control block of the user's login shell, or command interpreter. Unless modified, this user-identifier is inherited by all processes invoked from the initial login shell. Under certain conditions, the user-identifier may be determined with the system calls setuid () and getuid ().

The effective user-identifier is, by default, the same as the user-identifier, but may be temporarily changed to a different value to offer temporary privileges. The successful invocation of set-user-id programs, such as password and login will, typically, set the effective user-identifier for the lifetime of that process. Under certain conditions, the effective user-identifier may be changed and determined with the system calls seteuid () and geteuid ().

4.4.2 The Windows-NT Security Model

While the UNIX security model provides system-wide and consistent support of user and group identification, it constrains their manipulation to the system administrator (root).

In contrast, the newer Windows-NT security model enables each authorized user (and process) to both examine and manipulate access to a variety of objects. Again, the access controls provided by Windows-NT are best seen by examining the file system — but we must be using a partition supporting the Windows-NT File System (NTFS) and not simply a FAT-based (ala. Windows'98) partition.

4.5 Security in Operating Systems

It is the protection of operating system from theft or damage to the hardware. Side by side it is the software and to the information on them as well as from disruption or misdirection of the services they provide. It includes controlling physical access to the hardware as well as protecting against harm that may come via network access, data and code injection due to malpractice by operators , whether intentional, accidental to deviating from secure procedures.

The field is of growing importance due to the increasing reliance on computer systems and the Internet in most societies, wireless networks such as Bluetooth and Wi-Fi - and the growth of "smart" devices, including smart phones, televisions and tiny devices as part of the Internet of Things.

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.

Developing secure operating systems involves four activities. First, the environment to be protected must be well understood. Through policy statements and models, the essential components of systems are identified, and the interactions among components can be studied.

4.6 Operating System Security

We study operating systems in depth because they are at the heart of security systems for modern computers. They must provide mechanisms for both separation and sharing, mechanisms that must be robust and yet easy to use.
Physical Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc.
To ensure operating system security the following mechanisms are employed.

4.6.1 Authentication

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways −

- **Username / Password** − User need to enter a registered username and password with Operating system to login into the system.
- **User card/key** − User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
- **User attribute - fingerprint/ eye retina pattern/ signature** − User need to pass his/her attribute via designated input device used by operating system to login into the system.

4.6.2 One Time passwords

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password is implemented in various ways.

- **Random numbers** − Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** − User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** − Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

4.6.3 Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common examples of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generally a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user.

4.6.4 System Threats

A system threat refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. A system threat creates such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worm's processes can even shut down an entire network.

- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.

- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

4.6.5 Computer Security Classifications

As per the U.S. Department of Defense Trusted Computer System's Evaluation Criteria there are four security classifications in computer systems: A, B, C, and D. There is a widely used specification to determine and model the security of systems and of security solutions. Following is the brief description of each classification.

4.6.5.1 Classification Type & Description

Type A

Verified Design: no additional features in an A1 system over a B3 system; rather there are formal procedures for the analysis of the design of the system and more rigorous controls on its implementation. Highest Level uses formal design specifications and verification techniques and grants a high degree of assurance of process security.

Type B

This type provides a mandatory protection system. Have all the properties of a class C2 system. It attaches a sensitivity label to each object. It is of three types.

B1 − Labeled Security Protection: the system must implement the Mandatory Access Control in which every subject and object of the system must maintain a security label, and every access to system resource (objects) by a subject must check for security labels and follow some defined rules. It maintains the security label of each object in the system. This Label is used for making decisions to access control.

B2 − Structured Protection: few new security features are added beyond B1; rather the focus is on the structure (design) of the system to maintain greater levels of assurance so that the system behaves predictably and correctly (such as, a minimal security kernel, trusted path to user, and identified covert channels, etc). Extends the sensitivity labels to each system resource, such as storage objects, supports covert channels and auditing of events.

B3 − Security Domains: more requirements to maintain greater assurance that the system will be small enough to be subjected to analysis and tests, and not to have bugs that might allow something to circumvent mandatory access controls, e.g., support of active audit, and secure crashing, etc. Allows creating lists or user groups for access-control to grant access or revoke access to a given named object.

Type C

It provides protection and user accountability using audit capabilities. It is of two types.

C1 − Discretionary Security Protection: the system must identify different users (or jobs) running inside the system, and provide mechanisms for user authentication and authorization to prevent unprivileged user programs from interference of each other (e.g., overwriting critical portions of the memory). It incorporates controls so that users can protect their private information and keep other users from accidentally reading / deleting their data. UNIX versions are mostly Cl class.

C2 − Controlled Access Protection: the system meets additional security requirements than that of C1 that include access control at a per user granularity (access control for any subset of the user community); clearing of newly allocated disk space and memory; and ability of auditing (logging) for security relevant events such as authentication and object access, etc. It adds an individual-level access control to the capabilities of a Cl level system.

Type D

Minimal Protection: For this type no security is required; the system did not qualify for any of the higher ratings. This is a lowest level and minimum protection. MS-DOS, Window 3.1opearting systems fall in this category.

4.7 System access Threats

4.7.1 Intruders

Masquerader an individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account Misfeasor a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges Clandestine user an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

4.7.2 Malicious Software

Programs that exploit vulnerabilities in computing systems also referred to as malware. These can be divided into two categories: parasitic fragments of programs that cannot exist independently of some actual application program, utility, or system program viruses, logic bombs, and backdoors are examples independent self-contained programs that can be scheduled and run by the operating system worms and boot programs are examples.

4.7.3 Access Control

Implements a security policy that specifies who or what may have access toν each specific system resource and the type of access that is permitted in each instance Mediates between a user and system resources, such as applications,ν operating systems, firewalls, routers, files, and databases. A security administrator maintains an authorization database that specifiesν what type of access to which resources is allowed for this user the access control function consults this database to determine whether toν grant access an auditing function monitors and keeps a record of user accesses to systemν resources

4.7.4 Firewalls

Can be an effective means ofν protecting a local system or network of systems from network-based security threats while affording access to the outside world via wide area networks and the Internet Traditionally, a firewall is aν dedicated computer that interfaces with computers outside a network and has special security precautions built into it in order to protect sensitive files on computers within the network Design goals:

1) The firewall acts as a choke point, so that all incoming traffic and all outgoing traffic must pass through the firewall

2) The firewall enforces the local security policy, which defines the traffic that is authorized to pass

3) The firewall is secure against attacks

4.8 Requirements of Secure Operation Systems

Most current operating systems provide discretionary access control, that is, someone who owns a resource can make a decision as to who is allowed to use (access) the resource. Moreover, because the lack of built-in mechanisms for the enforcement of security policies in such systems, the access control is normally a one-shot approach: either all or none privileges are granted, rarely supporting the “principle of least privilege” (without limiting the privileges a program can inherit based on the trustworthiness).

Systems with stronger security and protection will require evolving from the approach of discretionary control towards the concept of mandatory (non-

discretionary) control where information is confined within a “security perimeter” with strict rules enforced by the system about who is allowed access to certain resources, and not allow any information to move from a more secure environment to a less secure environment.

A secure architecture requires flexibility for support of a wide variety of security policies: Separation of security policy logic from the mechanism of policy enforcement, so that a system can support diverse security policies. Support for policy definition and policy changes with well-defined policy interfaces and formats provide the default security behavior of the system so as to maintain tight system security without requiring detailed system configuration.

4.9 Design Principles of Secure Operation Systems

Saltzer and Schroeder (1975) identified a core set of principles to operating system security design:

Least privilege: Every object (users and their processes) should work within a minimal set of privileges; access rights should be obtained by explicit request, and the default level of access should be “none”.

Economy of mechanisms: security mechanisms should be as small and simple as possible, aiding in their verification. This implies that they should be integral to an operating system’s design, and not an afterthought.

Acceptability: security mechanisms must at the same time be robust yet non-intrusive. An intrusive mechanism is likely to be counter-productive and avoided by users, if possible.

Complete: Mechanisms must be pervasive and access control checked during all operations — including the tasks of backup and maintenance.

Open design: An operating system’s security should not remain secret, nor be provided by stealth. Open mechanisms are subject to scrutiny, review, and continued refinement.

4.10 Trusted System

Before we begin to examine a trusted operating system in detail, let us look more carefully at the terminology involved in understanding and describing trust. What would it take for us to consider something secure? The word secure reflects a dichotomy: Something is either secure or not secure. If secure, it should withstand all attacks, today, tomorrow, and a century from now. And if we claim that it is secure, you either accept our assertion (and buy and use it) or reject it (and either do not use it or use it but do not trust it). How does security differ from quality? If we claim that something is good, you are less interested in our claims and more interested in an objective appraisal of whether the thing meets your performance and functionality needs

For this reason, security professionals prefer to speak of trusted instead of secure operating systems. A trusted system connotes one that meets the intended security requirements, is of high enough quality, and justifies the user's confidence in that

quality. That is, trust is perceived by the system's receiver or user, not by its developer, designer, or manufacturer. As a user, you may not be able to evaluate that trust directly. You may trust the design, a professional evaluation, or the opinion of a valued colleague. But in the end, it is your responsibility to sanction the degree of trust you require.

4.10.1 Trusted Operating System Design

Operating systems by themselves (regardless of their security constraints) are very difficult to design. They handle many duties, are subject to interruptions and context switches, and must minimize overhead so as not to slow user computations and interactions. Adding the responsibility for security enforcement to the operating system substantially increases the difficulty of designing an operating system.

Nevertheless, the need for effective security is becoming more pervasive, and good software engineering principles tell us that it is better to design the security in at the beginning than to shoehorn it in at the end. Thus, this section focuses on the design of operating systems for a high degree of security. First, we examine the basic design of a standard multipurpose operating system. Then, we consider isolation, through which an operating system supports both sharing and separating user domains.

4.10.2 Assurances in Trusted Operating Systems

We began by studying different models of protection systems. By the time we reached in the last section, we examined three principles—isolation, security kernel, and layered structure—used in designing secure operating systems, and we looked in detail at the approaches taken by designers of particular operating systems. Now, we suppose that an operating system provider has taken these considerations into account and claims to have a secure design. It is time for us to consider assurance, ways of convincing others that a model, design, and implementation are correct.

4.11 Operating System Vulnerability

In computer security, vulnerability is a weakness which allows an attacker to reduce a system's information assurance. Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw. To exploit vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness. In this frame, vulnerability is also known as the attack surface.

Vulnerability management is the cyclical practice of identifying, classifying, remediating, and mitigating vulnerabilities. This practice generally refers to software vulnerabilities in computing systems.

A security risk may be classified as vulnerability. The use of vulnerability with the same meaning of risk can lead to confusion. The risk is tied to the potential of a significant loss. Then there are vulnerabilities without risk: for example when the affected asset has no value. A vulnerability with one or more known instances

of working and fully implemented attacks is classified as an exploitable vulnerability — a vulnerability for which can exploit exists. The window of vulnerability is the time from when the security hole was introduced or manifested in deployed software, to when access was removed, a security fix was available/deployed, or the attackers was disabled.

i. Most Common Operating Systems Vulnerabilities

We will discuss some of the operating system security vulnerabilities as follows:

a) File and share permissions that give up everything to everyone

This is easily the biggest vulnerability we are seeing with OS (Operating Systems) regardless of the type of system or Windows version. Users who create shares to make their local files available across the network are typically the culprits. Sometimes it's careless admins; other times they're honest mistakes. Unfortunately, all too often the "Everyone group" is given full access to every file on the system. Then, all it takes is for an insider to search for sensitive keywords stored in .pdf, .xls, .doc and other file formats using a text search tool such as Effective File Search or File Locator Pro. Odds are - nearly 100% of the time -- the attacker will come across sensitive information (SSNs, credit card numbers, you name it) that they shouldn't have access to. Best case scenario, this is an identity theft in the making. Worst case, this becomes a serious breach that makes the headlines.

b) Lack of malware protection

I know, it's really basic but I'm seeing it more now than ever. I've seen antivirus and antispyware software both disabled and not installed at all with no one being aware of the problem.

c) Lack of personal firewall protection

This is another basic security control that's still not enabled on many Windows systems. Even the basic (and free) Windows Firewall can prevent connections to the IPC$ and ADMIN$ shares that are often open and providing information and access that they shouldn't be divulging. Personal firewalls can also block malware infiltrations, wireless intrusions and more. I can't think of a good reason not to use a personal firewall on all workstations and most servers.

d) Weak or nonexistent drive encryption

The drive encryption marketing machine is working its magic, but I'm still seeing the majority of organizations (large and small) not using encryption. We believe that whole-disk encryption is the only way to go. If a laptop or desktop machine is lost or stolen, the only way to prevent someone from cracking the Windows password and gaining full access to the hard drive is to encrypt everything using reasonable passphrases. Relying on Windows Encrypted File System (EFS) or other file/directory/volume-level encryption puts too much security control in the hands of users and is a breach waiting to happen.

e) No minimum security standards

Users with wireless networks, especially, need to follow secure company policies at their homes, like requiring SSL for Outlook Web Access, a PPTP VPN

connection for remote network connectivity or WPA-PSK with a strong passphrase to help ensure everything is safe and sound. This can be tough to enforce without a workstation-based wireless IDS/IPS (typically a component of an enterprise wireless management system) or a well-configured Network Access Control (NAC) system. Nevertheless, make it your policy and enforce it wherever possible.

f) Missing patches in OS as well as third-party software, such as VNC, RealPlayer and others

This is a big problem that often gets overlooked. I'm not saying you should try to find these types of holes just to claim that patches are missing. Using Metasploit or its commercial alternatives CANVAS and CORE IMPACT, many missing patches can actually be exploited by a rogue insider or outsider who's gotten into your network via other means. Full remote access anyone?

g) Weak Windows security policy settings

Some examples of this include audit logging that is not being enabled for failed events; no password-protected screensavers; not requiring Ctrl+ Alt+ Del for login; not requiring password complexity; and displaying the last user name that logged in. Polices to control these issues are easy to implement locally on each Windows system for smaller Windows shops not running Active Directory. It's even easier for larger enterprises via Active Directory Group Policy.

h) Unaccounted for systems running unknown, and unmanaged, services such as IIS and SQL Server Express

These are often legacy Windows systems that aren't within the scope of enterprise security and compliance. Sometimes, they're not even supported by third-party security management apps so they get pushed aside. These systems (typically Windows 98, NT and 2000) are often unhardened and un-patched and are waiting to be exploited. Inevitably there's going to be some random training or test system that everyone forgot about. But such a system is all it takes for someone with ill intent to get onto your network and do bad things.

i) Weak or nonexistent passwords

We can't tell you how many systems (especially Windows laptops) we see that do not have a password assigned to the Administrator account or the default user's password is the same as the user name. The password problem has been around since the dawn of time, so there's no excuse for this one.

j) Windows Mobile and other mobile device weaknesses

In today's mobile world, we would be remiss to not at least mention the vulnerabilities associated with Windows Mobile and similar mobile devices. Some mobile-specific issues are essential to have on your radar. In a tip called Windows mobile security: Get it locked down, we outline several things to consider as follows.

4.12 Remove Unnecessary Services, Applications and Protocols

The system planning process should identify what is actually required for a given system so that a suitable level of functionality is provided, while eliminating software that is not required to improve security When performing the initial installation the supplied defaults should not beν used, but rather the installation should be customized so that only the required packages are installed Many of the security-hardening guides provide lists of services, applications, and protocols that should not be installed if not required Strong preference is stated for not installing unwanted software, rather than installing and then later removing or disabling it as many uninstall scripts fail to completely remove all components of a package should an attacker succeed in gaining some access to a system, disabled software could be re-enabled and used to further compromise a system it is better for security if unwanted software is not installed, and thus not available for use at all.

4.13 Install Additional Security Controls

Further security improvement mayν be possible by installing and configuring additional security tools such as antivirus software, host-based firewall, IDS or IPS software, or application white listing Some of these may be supplied as part of the operating systems installation, but not configured and enabled by default Given the wide-spread prevalence of malware, appropriate antivirus is a critical security component IDS and IPS software may include additional mechanisms such as traffic monitoring or file integrity checking to identify and even respond to some types of attack White-listing applications limits the programs that can execute in the system to just those in an explicit list.

4.14 Test the System Security

The final step in the process of initially securing the base operating system isν security testing The goal is to ensure that the previous security configuration steps areν correctly implemented and to identify any possible vulnerabilities that must be corrected or managed Suitable checklists are included in many security-hardening guidesν There are also programs specifically designed to review a system to ensureν that a system meets the basic security requirements and to scan for known vulnerabilities and poor configuration practices This should be done following the initial hardening of the system and thenν repeated periodically as part of the security maintenance process

Logging

Effective logging helps ensure that in the event of a system breach or failure, system administrators can more quickly and accurately identify what happened and more effectively focus their remediation and recovery efforts Logging information can beν generated by the system, network, and applications The range of logging datαν acquired should be determined during the system planning

stage Logging can generate significant volumes of information so it is important that sufficient space is allocated for them A suitable automatic log rotationν and archive system should be configured to assist in managing the overall size of the logging information Some form of automated analysisν is preferred as it is more likely to identify abnormal activity manual analysis of logs isν tedious and is not a reliable means of detecting adverse events

4.15 Protection in General-Purpose Operating Systems

We looked at several types of security problems that can occur in programs. The problems may be unintentional, as with buffer overflows, or intentional, as when a virus or worm is inserted in code.

In addition to these general problems, certain kinds of programs may be vulnerable to certain kinds of security problems simply because of the nature of the program itself.

For example, operating systems and databases offer security challenges beyond those in more general programs; these programs offer different access to different items by different kinds of users, so the program designers must pay careful attention to defining access, granting access, and controlling intentional and unintentional corruption of data and relationships.

4.16 Let us Sum-up

The approach covered in this unit – executing applications from a strongly guarded, secure operating system – certainly opens an alternative frontier in battling with many of existing cyber-space threats of the real world. Although, the approach of using secure operating systems will not be a panacea for all the dangers of current cyber space, and the security of individual applications may still suffer from the vulnerabilities of their own, with the strong containment of a secure operation system, the damages caused from a compromise within one application would be much localized, and the impacts among various applications could be much well controlled.

4.17 Self Assessment Questions

1. What is a security model? What are the uses of Operating System security model?

   ……
   ……
   ……
   ……
2. Define Operating System Security. What are the mechanisms employed to ensure operating system security?

   ……
   ……

……
……

3. What is a Program threat? List and explain some well-known program threats.

……
……
……
……

4. What do you mean by System Threats? Explain different system threats?

……
……
……
……

5. What do you mean by secured operating system? Write the requirements of Secured Operation Systems.

……
……
……
……

6. Explain the design principles of Secure Operation Systems?

……
……
……
……

7. What do you mean by Operating System Vulnerability? Name the most Common Operating Systems Vulnerabilities.

……
……
……
……

4.18 References and Further Readings

1. Cui-Qing Yang, Operating System Security and Secure Operating Systems, Version 1.4b, Option 1 for GSEC January 2003.
2. www.av-Comparatives .org
3. Stewart, W. (2000, Jan. 07). Email Security. Retrieved Feb. 17, 2016,
4. Bose, R. (2008). McGraw Hill Education. McGraw Hill Education.
5. https://en.wikipedia.org/wiki/Vulnerability_(computing)

Answer to Self-assessment Questions (Unit-1)

1. **What is a desktop computer?**

   A **desktop computer** is a personal computer designed for regular use at a single location or near a desk or table due to its size and power requirements. The most common configuration has a cabinet that encloses the power supply, motherboard with a microprocessor as the central processing unit (CPU), memory, bus, and other electronic components, disk storage usually hard disk, drives, optical disc drives, a keyboard and a mouse for input; computer monitor and additionally a printer for output. An all-in-one desktop computer typically combines the CPU cabinet and monitor in one unit.

2. **What are the desktop security threats?**

   Desktop systems face a number of vulnerabilities and security threats. Desktop is the entry point to the organization's information resources. If the security of the desktop is weak, potential intruders can easily by-pass the first obstacle.
   Some of the threats that need to be constantly communicated to desktop users are:

   a) Using programs that send password to the intended person.
   b) Bad password management, weak password, sharing password, never change password, “post-it note” habits etc.
   c) Guest accounts or open accounts.
   d) Social engineering attacks.
   e) Virus and other malicious code of attacks.
   f) Unsolicited email attachments.
   g) Downloading software from un-trusted Internet sites.
   h) Installing software from un-trusted sources.
   i) Modem connection to desktop within the LAN thus creating a backdoor.
   j) Unattended desktop, without screen lock.
   k) Packet sniffing.
   l) Bad desktop management, no anti-virus, outdated virus signature, no backups, no desktop lock, opens folder shares without password, opening insecure access etc.

3. **Mention the desktop security policies?**

A basic list of the security policies would include an anti-virus program, use of licensed and Open source Software, Software updates, Anti-spyware, a personal firewall, file-encryption software etc. as security solutions.

Each one of these mechanisms will help ensure at least one or more of the three basic principles of security, Confidentiality, Integrity and Availability are addressed.

4. **What are the tips to make your personal computer secured?**

The basic steps for securing one's desktop system or personal computer are as follows:

a) Keep operating system patches up to date.
b) Use encryption to securely encode sensitive information
c) Install antivirus software; configure for daily updates
d) Install and configure a personal firewall
e) Keep application and software patches up to date (e.g., Microsoft Office, browsers, etc.)
f) Follow best practices when opening email attachments
g) Follow secure password policies
h) Follow best practices for user account security
i) Eliminate unnecessary network services, applications, and processes
j) Avoid peer-to-peer file sharing
k) Install and configure anti-Spyware programs
l) Configure system restore points to protect your current configuration
m) Perform regularly scheduled backups to protect data
n) Turn off computer when not in use; restrict physical access to computer.

5. **Mention the ways of ensuring physical security to a desktop.**

The simple way to desktop security from physical access is to either to set a password protected screen saver or, even better, to get into the habit of locking computers when users walk away from them. A screen saver timeout should be utilized so if a user walks away from their computer, the password-protected screen saver would come up.

Another item that could be addressed is to make sure users understand that it is important that they shut down their computers at the end of the day. Sometimes this allows for valuable updates to be applied and doing your own part for a greener environment. If somehow a potential attacker gains access to a computer that is turned off, they will be less likely to utilize it than one that is already turned on and unlocked.

Answer to Self-assessment Questions (Unit-2)

1. **What is a programming bug?**

   A programming bug is an error, flaw, failure or fault in a computer program that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. Most bugs arise from mistakes and errors made by people in either a program's source code or its design or compilers producing incorrect code. Example: Division by Zero Error.

2. **What are impacts of programming bugs?**

   The impact of programming bugs tends to vary and could have a wide range of impact on the software's end-user. Some are very simple, such as a word editing program that might take a little extra time loading. This is not detected easily for some time. Others are more serious and may cause the application crash when performing certain actions. In this scenario, one normally receives an error message briefly detailing the problem such and division by zero error. We may also have the worst type of programming bugs, which qualify as security vulnerabilities. Such errors are typically flaws with the operating systems or web browsers. Such deficiencies could open exploits for intruders and malicious software writers and can give them control of a system.

3. **How can you classify the programming bugs?**

   Programming bugs are classified into five broad categories as follows.

   a) Token Error

   A token error occurs whenever our program contains a word or symbol that is not in Java's vocabulary.

   b) Syntax Error

   Even though the Java compiler may recognize every token in a program, the program still may contain a syntax error. This type of error occurs whenever we use incorrect grammar or punctuation (according to the syntax rules of the Java programming language).

   c) Syntax Constraint Error

   These errors occur when the Java compiler cannot determine the meaning of a program. Sometimes a sentence might seem syntactically correct but, meaningless.

   All these errors are called **compile-time** errors, because the Java compiler detects them while compiling our programs. Errors that occur when the program is running (or executing) are called **run-time** errors. Since the compiler points out compile-time errors, they are much easier to fix.

   d) Execution Error

   Execution errors occur when the Java runtime system is executing a program and discover that it can't legally carry out one of our instructions (for example, division by 0). If it recognizes such a case, it terminates execution of the program (again, supplying some information about the error).

   e) Intent Error

An intent error occurs whenever Java successfully completes execution of a program, but the program doesn't compute the correct answer.

Example of Bug: TYPE - Accidental

for (i=0; i<numrows; i++)
for (j=0; j<numcols; j++);
pixels++;

The error caused by a stray ";" on line 2.

Such accidental bugs are often caused by stray characters, etc. While "minor" in their fix, they can be the devil to find!

4. **How can you be a zero- bug programmer?**

A good programmer should be able to ensure that the code he or she changes is reliable, correct, and thoroughly self-verified. One should completely understand all the results and impacts on changes will cause. The characteristics which will make a better programmer are as follows.

- Be humble -- you are and will be making mistakes. Repeatedly.
- Be fully aware of the limited size of your code.
- Get rid of changeable state (where ever possible). Learn functional programming.
- Reduce number of possible code paths
- Understand (the magnitude of) the size of the input & output spaces (of your functions) and try to reduce them in order to get ever closer to 100% test coverage
- Always assume your code is not working -- prove it otherwise!

5. **What is a malicious code?**

Malicious code refers to a broad category of programs that can cause damage or undesirable effects to computers or networks. Potential damage can include modifying, destroying or stealing data, gaining or allowing unauthorised access to a system, bringing up unwanted screens, and executing functions that a user never intended. Malicious code is an application security threat that cannot be efficiently controlled by conventional antivirus software.

6. **What are different malicious codes?**

Malicious code is the term used to describe any code in any part of a software system or script that is intended to cause undesired effects, security breaches or damage to a system. Different types of malicious code include computer viruses, worms, Trojan horses, logic bombs, spyware, adware and backdoor programs.

7. **What is a Trojan horse? What are different types of Trojan horse?**

Trojan horse is malicious software that can be hiding in a victim computer. Trojans do not have their own on-board replication and spreading capability. There are many ways for infecting victim computers by Trojans such as downloading from a remote site, but recently Trojans use worm and virus for penetrating into victim computers. There is special kind of Trojan which can be controlled remotely and receive commands from attackers. Trojans similar to worms can have approximately any consequence on the target hosts.

We categorize Trojan horse into two main groups:

- **General Trojans:** this type of Trojans has wide range of malicious activities. They can threaten data integrity of victim machines. They can redirect victim machines to a particular web site by replacing system files that contain URLs.
- **Remote-Access Trojans:** we can claim that they are the most dangerous type of Trojan. They have special capability which enable attacker to remotely control victim machine via a LAN or Internet. This type of Trojan can be instructed by attacker for malicious activities such as harvesting confidential information from the victim machine.

Answer to Self Assessment Questions (Unit-3)

1. What do you mean by database security? When do database security issues arise?

Database Security is the mechanism which protects the database against intentional or accidental threats. Database security issues arise due to the following situations.

- Theft and Fraud
- Loss of confidentiality
- Loss of privacy
- Loss of integrity
- Loss of availability

2. Name and explain the computer based database security counter measures.

There are some counter measures which are based on computer based controls. They are:

a) Authorization
b) Views
c) Backup and Recovery
d) Encryption
e) RAID Technology

a) Authorization

The granting of a privilege that enables a user to have a legitimate access to system is called authorization. It is sometimes referred as access control. The process of authorization involves authenticating the user requesting access to objects. A system administrator is responsible for allowing users to have access to the system by creating individual user accounts.

b) Views

A database view is a virtual relation that does not actually exist in the database, but is produced upon request by a particular user, at the time of request.

The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users. The user is not aware of the existence of any attributes or rows that are missing from the view.

c) Backup & Recovery

The process of periodically taking a copy of the database and log file on to offline storage media. DBMS should provide backup facilities to assist with the recovery of a database failure.

d) Encryption

The encoding of data by an encryption algorithm makes the data unreadable by any program without the decryption key. Thereby it protects the data transmitted over communication lines.

e) RAID (Redundant Array of Independent Disks)

The use of RAID technology works on having a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance.

3. Mention the requirements for database security.

The requirements for database security include the following.

a) **Physical database integrity**: The data of a database are immune to physical problems, such as power failures, and someone can reconstruct the database if it is destroyed through a catastrophe.

b) **Logical database integrity:** The structure of the database is preserved. With logical integrity of a database, a modification to the value of one field does not affect other fields, for example.

c) **Element integrity:** The data contained in each element are accurate.

d) **Audit ability:** It is possible to track who or what has accessed (or modified) the elements in the database.

e) **Access control:** A user is allowed to access only authorized data, and different users can be restricted to different modes of access (such as read or write).

f) **User authentication**: Every user is positively identified, both for the audit trail and for permission to access certain data.

g) **Availability:** Users can access the database in general and all the data for which they are authorized.

4. **Discuss how consistency is maintained by concurrent execution of database transactions.**

Database systems are often multiuser systems. Accesses by two users sharing the same database must be constrained so that neither interferes with the other. Simple locking is done by the DBMS. If two users attempt to read the same data item, there is no conflict because both obtain the same value. But if multiple users access the same database at the same time (concurrently), inconsistencies may arise.

The following mechanisms are used to maintain consistencies in the databases.

a) Monitors

The monitor is the unit of a DBMS responsible for the structural integrity of the database. A monitor can check values being entered to ensure their consistency with the rest of the database or with characteristics of the particular field. For example, a monitor might reject alphabetic characters for a numeric field.

b) Range Comparisons

A range comparison monitor tests each new value to ensure that the value is within an acceptable range. If the data value is outside the range, it is rejected and not entered into the database

c) State Constraints

State constraints describe the condition of the entire database. At no time should the database values violate these constraints. Phrased differently, if these constraints are not met, some value of the database is in error.

d) Transition Constraints

State constraints describe the state of a correct database. Transition constraints describe conditions necessary before changes can be applied to a database. For example, before a new employee can be added to the database, there must be a position number in the database with status "vacant."

e) **Sensitive Data**

Some databases contain what is called sensitive data. As a working definition, let us say that sensitive data are data that should not be made public. Determining which data items and fields are sensitive depends both on the individual database and the underlying meaning of the data

5. **As a database security profession suggest the points to keep your database Secured.**

The following tips are suggested to keep a database secured.

a) Enable Security Controls

Make sure you check the security controls and enable all of the features before allowing anyone access to the database.

b) Check the Patch Level

Check the patch level configuration in the database to determine if there is any vulnerability in the default settings. Also, perform a full assessment of the database to fix any existing vulnerabilities in the system before placing any data into the database.

c) Exclude copying of the Database

You should not allow database copying because it represents an internal threat to database security.

d) Restrict Access

Restrict access to the database by specifically designating who is allowed administrator privileges. For a small business it is a good idea to delegate this responsibility to one IT administrator and then place certain restrictions on other users.

e) Existing Databases

There are database discovery tools which identify existing databases that contain confidential information. The tools also monitor existing databases to ensure the information is stored in encrypted format. In addition to the new database, make sure you monitor all of the existing databases to ensure that information is encrypted, there are no vulnerabilities, and that there are no duplicates.

f) Shared Data

The organization has to train new employees and developers have to test new database applications. The IT administrator can perform sub-setting which provides a separate type of restricted access with fake information substituted for the sensitive information. Sub setting a database basically allows developers and new employees to use the database for testing and training without exposing confidential or sensitive information.

6. What do you mean by sensitive data? Mention the factors that can make the data sensitive.

Some databases contain sensitive data. The sensitive data are data that should not be made public. Determining which data items and fields are sensitive depends both on the individual database and the underlying meaning of the data. Several factors can make data sensitive-:

- **Inherently sensitive.** The value itself may be so revealing that it is sensitive. Examples are the locations of defensive missiles or the median income of barbers in a town with only one barber.

- **From a sensitive source.** The source of the data may indicate a need for confidentiality. An example is information from an informer whose identity would be compromised if the information were disclosed.

- **Declared sensitive.** The database administrator or the owner of the data may have declared the data to be sensitive. Examples are classified military data or the name of the anonymous donor of a piece of art.

- Part of a sensitive **attribute** or a sensitive **record**. In a database, an entire attribute or record may be classified as sensitive. Examples are the salary attribute of a personnel database or a record describing a secret space mission.

- Sensitive **in relation to previously disclosed information**. Some data become sensitive in the presence of other data. For example, the longitude coordinate of a secret gold mine reveals little, but the longitude coordinate in conjunction with the latitude coordinate pinpoints the mine.

7. What is an SQL injection? How SQL injection must exploit security vulnerability in an application's software.

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).

SQL injection exploits a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.

SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

Answer to Self assessment Questions (Unit-4)

1. **What is a security model? What are the uses of Operating System security model?**

   A **security** model is a representation of the **security policy** for the OS. A models used to describe, study, or analysis of a particular situation or relationship. Security models are used to

   i) test a particular policy for completeness and consistency

   ii) document a policy

   iii) help conceptualize, design and implementation

   iv) check whether an implementation meets its requirements

2. **Define Operating System Security. What are the mechanisms employed to ensure operating system security?**

   The operating system provides an interface to the underlying hardware and data. It acts as a platform on which various applications execute their operations. 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.

   If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. To ensure operating system security the following mechanisms are employed.

   a) Authentication

   Authentication refers to identifying each user of the system and associating the executing programs with those users. Operating Systems generally identifies/authenticates users using following three ways −

   - Username / Password
   - User card/key
   - User attribute - fingerprint/ eye retina pattern/ signature

b) One Time passwords

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password is implemented in various ways.

- Random numbers
- Secret key
- Network password

c) Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**.

Following is the list of some well-known program threats.

- Trojan Horse
- Trap Door
- Logic Bomb
- Virus

- **System Threats**

A system threat refers to misuse of system services and network connections to put user in trouble. Following is the list of some well-known system threats.

- Worm
- Port Scanning
- Denial of Service.

3. **What is a Program threat? List and explain some well-known program threats.**

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common examples of program threat is a program installed in a computer which can store and send user credentials via

network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generally a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

4. What do you mean by System Threats? Explain different system threats?

A system threat refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. A system threat creates such an environment that operating system resources/ user files are misused.

Following is the list of some well-known system threats.

- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worm's processes can even shut down an entire network.
- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

5. What do you mean by secured operating system? Write the requirements of Secured Operation Systems.

A secure operating system also referred to as a "trusted OS." The primary objective of both of these is to provide a secure operating environment; each takes a different approach for meeting this objective.

A trusted operating system is that has to be locked down to prevent attacks, a trusted operating system manages data to make sure that it cannot be altered or moved and that it can be viewed only by persons having appropriate and authorized access rights.

A secure architecture requires flexibility for support of a wide variety of security policies: Separation of security policy logic from the mechanism of policy enforcement, so that a system can support diverse security policies. Support for policy definition and policy changes with well-defined policy interfaces and formats provide the default security behavior of the system so as to maintain tight system security without requiring detailed system configuration.

6. **Explain the design principles of Secure Operation Systems?**

A set of principles to operating system security design are as follows:

Least privilege: Every object (users and their processes) should work within a minimal set of privileges; access rights should be obtained by explicit request, and the default level of access should be "none".

Economy of mechanisms: security mechanisms should be as small and simple as possible, aiding in their verification. This implies that they should be integral to an operating system's design, and not an afterthought.

Acceptability: security mechanisms must at the same time be robust yet non-intrusive. An intrusive mechanism is likely to be counter-productive and avoided by users, if possible.

Complete: Mechanisms must be pervasive and access control checked during all operations — including the tasks of backup and maintenance.

7. **What do you mean by Operating System Vulnerability? Name the most Common Operating Systems Vulnerabilities**.

In computer security, vulnerability is a weakness which allows an attacker to reduce a system's information assurance. Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw. To exploit

vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness. In this frame, vulnerability is also known as the attack surface.

Most Common Operating Systems Vulnerabilities

a) File and share permissions that give up everything to everyone

b) Lack of malware protection

c) Lack of personal firewall protection

d) Weak or nonexistent drive encryption

e) No minimum security standards

f) Missing patches in OS as well as third-party software, such as VNC, RealPlayer and others

g) Weak Windows security policy settings

h) Unaccounted for systems running unknown, and unmanaged, services such as IIS and SQL Server Express

i) Weak or nonexistent passwords

j) Windows Mobile and other mobile device weaknesses