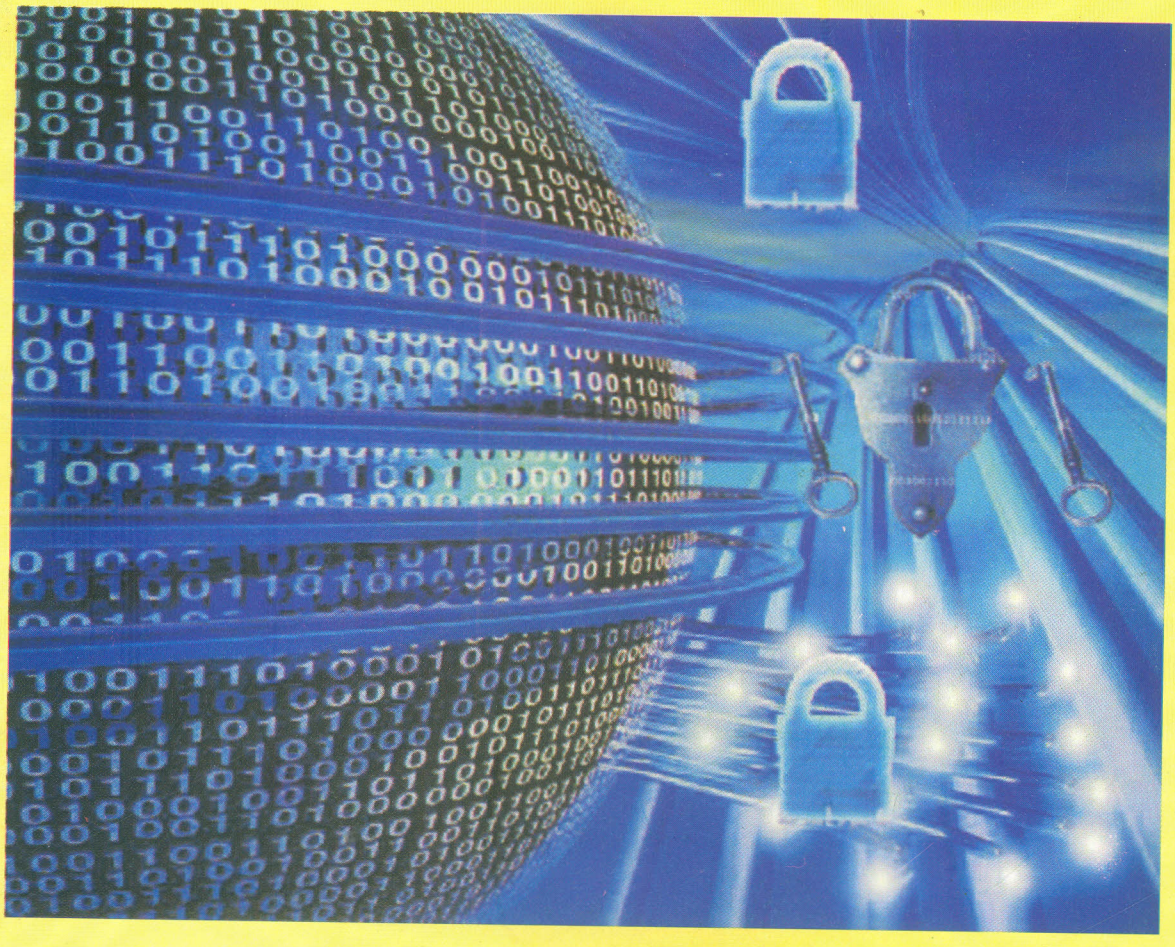
ignou
THE PEOPLE'S
UNIVERSITY

Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-025
Application and Business Security Developments



Secure Application Development-II

3

"शिक्षा मानव को बन्धनों से मुक्त करती है और आज के युग में तो यह लोकतंत्र की भावना का आधार भी है। जन्म तथा अन्य कारणों से उत्पन्न जाति एवं वर्गगत विषमताओं को दूर करते हुए मनुष्य को इन सबसे ऊपर उठाती है।"

– इन्दिरा गांधी

"Education is a liberating force, and in our age it is also a democratising force, cutting across the barriers of caste and class, smoothing out inequalities imposed by birth and other circumstances."

—Indira Gandhi

Indira Gandhi National Open University
School of Vocational Education and Training

MSEI-025
Application and Business Security Developments

Block

3

Secure Application Development -II

Programme Expert/ Design Committee of Post Graduate Diploma in Information Security (PGDIS)

Prof. K.R. Srivathsan
Pro Vice-Chancellor, IGNOU

Mr. B.J. Srinath, Sr. Director & Scientist 'G',CERT-In, Department of Information Technology, Ministry of Communication and Information Technology Govt of India

Mr. A.S.A Krishnan, Director, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Mr. S. Balasubramony, Dy. Superintendent of Police, CBI, Cyber Crime Investigation Cell, Delhi

Mr. B.V.C. Rao, Technical Director, National Informatics Centre, Ministry of Communication and Information Technology

Prof. M.N. Doja, Professor, Department of Computer Engineering, Jamia Milia Islamia
New Delhi

Dr. D.K. Lobiyal, Associate Professor, School of Computer and Systems Sciences, JNU
New Delhi

Mr. Omveer Singh, Scientist, CERT-In, Department of Information Technology, Cyber-Laws and E-Security Group, Ministry of Communication and Information Technology, Govt of India

Dr. Vivek Mudgil, Director, Eninov Systems
Noida

Mr. V.V.Subrahmanyam, Assistant Professor
School of Computer and Information Science
IGNOU

Mr. Anup Girdhar, CEO, Sedulity Solutions & Technologies, New Delhi

Prof. A.K. Saini, Professor, University School of Management Studies, Guru Gobind Singh Indraprastha University, Delhi

Mr. C.S. Rao, Technical Director in Cyber Security Division, National Informatics Centre, Ministry of Communication and Information Technology

Prof. C.G. Naidu, Director, School of Vocational Education & Training, IGNOU

Prof. Manohar Lal, Director, School of Computer and Information Science, IGNOU

Prof. K. Subramanian, Director, ACIIL, IGNOU
Former Deputy Director General, National Informatics Centre, Ministry of Communication and Information Technology, Govt. of India

Prof. K. Elumalai, Director, School of Law
IGNOU

Dr. A. Murali M Rao, Joint Director, Computer Division, IGNOU

Mr. P.V. Suresh, Sr. Assistant Professor, School of Computer and Information Science, IGNOU

Ms. Mansi Sharma, Assistant Professor, School of Law, IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training, IGNOU
Programme Coordinator

Block Preparation

Unit Writers

Ms. Shalini Chawla
M.Tech (Computer Science), Assistant Professor, Northern India Engineering College (NIEC), (Affiliated to Guru Gobind Singh Indraprastha University) Delhi (Unit 1)

Dr. Neeru Mundra
Associate Professor, Banarsidas Chandiwala Institute of Professional Studies, Dwarka, New Delhi

Ms. Renu Vashisth
Assistant Professor, Banarsidas Chandiwala Institute of Professional Studies, Dwarka, New Delhi (Unit 2)

Ms. Dimpal Arora
Assistant Professor (Computer Science)
Institute of Innovation in Technology & Management, Janak Puri
New Delhi (Unit 3)

Mr. Kaushal Mehta
Assistant Professor
Bhai Parmanand of Business Studies, Govt. of NCT Delhi, Shakarpur
Delhi (Unit 4)

Block Editor

Mr. P.V. Suresh
Sr. Assistant Professor, School of Computer and Information Science, IGNOU

Ms. Urshla Kant
Assistant Professor, School of Vocational Education & Training
IGNOU

Proof Reading and Format Editing

Ms. Urshla Kant
Assistant Professor, School of Vocational Education and Training, IGNOU

PRODUCTION

Mr. B. Natrajan	Mr. Jitender Sethi	Mr. Hemant Parida
Dy. Registrar (Pub.)	Asstt. Registrar (Pub.)	Proof Reader
MPDD, IGNOU	MPDD, IGNOU	MPDD, IGNOU

Feb, 2012

ISBN: 978-81-266-5891-6

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110 068 or the website of IGNOU www.ignou.ac.in

Printed and Published on behalf of the Indira Gandhi National Open University, New Delhi, by the Registrar, MPDD.

Printed at: Berry Art Press A-9, Mayapuri, Phase-I New Delhi-64

BLOCK INTRODUCTION

This block deals with the secure application development-II. Application security encompasses measures taken throughout the application's life-cycle to prevent exceptions in the security policy of an application or the underlying system (vulnerabilities) through flaws in the design, development, deployment, upgrade, or maintenance of the application. Applications only control the use of resources granted to them and not which resources are granted to them. They, in turn, determine the use of these resources by users of the application through application security. This block comprises of four units and is designed in the following way;

The **Unit One** is an effort towards answering some of the fundamental queries about Data Access. There are various kinds of security controls-access, flow, cryptographic etc and all of them complement each other to provide the security to the important data. Management and reporting capabilities are critical in any data security solution deployment. All the preventive measures discussed here to subject to practical limitations which keep them away from achieving their objectives under all conditions. There is no such mechanism which is perfectly secure and provides 100% security against unauthorized data access but they are good enough to reduce the risk of compromise to an acceptable level.

The **Unit two** covers error handling and logging. The code written in an extendable fashion so that new handlers can be added at any time while the core implementation can also be changed without side effects.

Unit three covers server configuration and code management. The methodology used in this unit allows you to build a secure Web server from scratch and also allows you to harden the security configuration of an existing Web server. The next step is to ensure that any deployed applications are correctly configured. The unit focuses on the application server configuration and the associated communication channels that connect the Web server to the application server and the application server to the database server.

Unit four explains about the application threat modeling. The most important issues to be considered in security are privacy, authentication, integrity and non repudiation. Privacy can be obtained by encryption of the plain text into cipher text. Authentication, integrity and non repudiation are achieved by using the digital signature. It covers entity authentication protocol, key establishment and time stamping technique. Using the concept of time stamping an eye can be kept at the time of development of a document. So in order to maintain the security issues, different techniques can be used so that the document should be authorized received without any tempering.

Hope you benefit from this block.

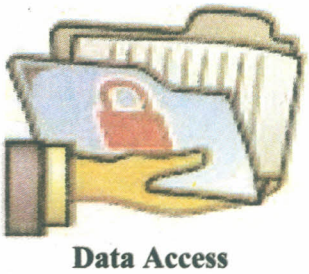
ACKNOWLEDGEMENT

The material we have used is purely for educational purposes. Every effort has been made to trace the copyright holders of material reproduced in this book. Should any infringement have occurred, the publishers and editors apologize and will be pleased to make the necessary corrections in future editions of this book.

UNIT 1 DATA ACCESS

Structure

1.0 INTRODUCTION



Data Access

Data access typically refers to software and activities related to storing, retrieving, or acting on data housed in a database or other repository. The database is a prime target for application level attacks. Application level attacks are used to exploit vulnerabilities in your data access code to gain access to the database. If all other attack vectors are closed, then the application's front door, port 80, becomes the path of choice for an attacker to steal, manipulate, and destroy data.

This unit will show how to secure data access and avoid common vulnerabilities and pitfalls. The unit also presents a series of countermeasures and defensive techniques that you can be used to mitigate the top threats related to data access.

1.1 OBJECTIVES

After studying this unit, you should be able to explain:

- details of various data security threats and countermeasures;
- the design guidelines for secure web applications;
- security importance for your database server;
- different methods that can be used to build secure assemblies; and
- preventive methods for unauthorized access.

1.2 DATA ACCESS AND SECURITY

This section will show you the top necessary actions and various business policies to secure the data access and will highlight the common vulnerabilities. Unless you use the correct countermeasures discussed in this unit, an attacker could exploit your data access code to run arbitrary commands in the database. Conventional security measures such as firewalls and SSL provide defense to some attacks but you should thoroughly validate your input and use parameterized stored procedures as a minimum defense.

1.2.1 The "Absolutely Necessary" Actions

Computer viruses are very common now days and tools are available to detect them. Can you find them?

Various actions must be taken to provide basic information security for your information, computers, and networks. Following list mentions few of them:

The data must be protected from damage by viruses, spyware, and other malicious code.

You can Install, use (in "real-time" mode, if available), and keep regularly updated anti-virus and anti-spyware software on every computer used in your business/organization.

Many commercial software vendors provide adequate protection at a reasonable price and some for free. Most vendors as well now a day's offer subscriptions to "security service" applications, which provide multiple layers of protection (in addition to anti-virus and anti-spyware protection).

Security Must Be Provided For Your Internet Connection

Most businesses/organizations have broadband (high speed) access to the Internet. It is important to keep in mind that this type of Internet access is always "on." Therefore, your computer - or any network your computer is attached to - is exposed to threats from the Internet so it is very critical to install and keep operational a hardware firewall between your internal network

and the Internet. This may be a function of a wireless access point/router or may be a function of a router provided by the Internet Service Provider (ISP).

Install and Activate Software Firewalls on All Your Systems

You should install, use, and keep updated a software firewall on each computer system used in your business/organization. For example:

If you use the Microsoft Windows operating system, it probably has a firewall included. You have to ensure that the firewall is operating, but it should be available.

Firewall is designed to permit or deny network transmissions based upon a set of rules.

To check the software firewall provided with Microsoft Windows XP, click on "Start" then "Settings", then "Control Panel", then "Windows Firewall". Select the "General" tab on the top of the popup window. You can see if the firewall is on or off. If it is off, select "On-Recommended" in the hollow circle next to the green check-mark icon.

To check the software firewall provided with Microsoft Windows Vista, click on "Start" then "Control Panel" then "Windows Firewall." If your firewall is working, you should see a message that "Windows Firewall is helping to protect your computer." If not, click on 'Turn Windows Firewall on or off" (in the upper left corner of the window) and select "Turn on firewall."

When using other commercial operating systems, ensure that you fully review operations manuals to discover if your system has a firewall included and how it is enabled.

Make Backup Copies of Important Business Data/Information

You should back up your data on each computer used in your business/organization. Your data includes (but is not limited to) word processing documents, electronic spreadsheets, databases, financial files, human resources files, accounts receivable/payable files, and other information used in or generated by your business/organization.

It is necessary to back up your data because computers die, hard disks fail, employees make mistakes, and malicious programs can destroy data on computers. Without data backups, you can easily get into a situation where you have to recreate your business data from paper copies and other manual files.

How Encryption is useful for business? Find Out.

Control Physical Access to Your Computers and Network Components

Do not allow unauthorized persons to have physical access to or to use of any of your business computers. This includes locking up laptops when they are not in use. It is a good idea to position each computer's display (or use a privacy screen) so that people walking by cannot see the information on the screen.

Secure Your Wireless Access Point and Networks

If you use wireless networking, it is a good idea to set the wireless access point so that it does not broadcast its Service Set Identifier (SSID). Also, it is critical to change the administrative password that was on the device when you received it. It is important to use strong encryption so that your data being transmitted between your computers and the wireless access point cannot be easily intercepted and read by electronic eavesdroppers. The current recommended encryption is Wi-Fi Protected Access 2 (WPA-2) – using the Advanced Encryption Standard (AES) for secure encryption.

1.2.2 Business Policies Related To Information Security

By implementing the best practices and other business policies related to the security of the data access you market your business as one in which the safety and security of your customer's information is of highest importance.

Every business needs written policies to identify acceptable practices and expectations for business operations.

Some policies will be related to human resources, others will relate to expected employee practices for using business resources, such as telephones, computers, printers, fax machines, and Internet access. This is not an exhaustive list and the range of potential policies is largely determined by the type of business and the degree of control and accountability desired by management. Legal and regulatory requirements may also require certain policies to be put in place and enforced.

Policies for information, computer, network, and Internet security, should communicate clearly to everyone. These policies should identify those information and other resources which are important to management and should clearly describe how management expects those resources to be used and protected by all.

For example, for sensitive employee information a typical policy statement might say, "All employee personnel data shall be protected from viewing or changing by unauthorized persons." This policy statement identifies a particular type of information and then describes the protection expected to be provided for that information.

Policies should be communicated clearly to each everyone and all should sign a statement agreeing that they have read the policies, that they will follow the policies, and that they understand the possible penalties for violating those policies. This will help management to hold the person accountable for violation of the businesses policies. As noted, there should be penalties for disregarding policies. And, those penalties should be enforced fairly and consistently for everyone in the business/organization that violates the policies.

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Is it safe to send sensitive information over the Internet?

.....

.....

.....

.....

2) Explain the advantages of implementing the policies related to information security?

.....

.....

.....

.....

3) What's the difference between a threat, vulnerability, and a risk?

.....

.....

.....

.....

4) How Spyware are dangerous threat than viruses. What makes spyware so destructive explain?

.....

.....

.....

.....

1.4. DATA ACCESSING IN DIFFERENT FRAMEWORKS

ACCESSING DATA IN .NET FRAME WORK

There are a plethora of ways to access SQL Server and other databases from .NET. All have their pros and cons and it will never be a simple question of which is "best" – the answer will always be "it depends".

1) "Standard" ADO.NET

ADO.NET is the data access model for .NET-based applications. It can be used to access relational database systems such as SQL Server 2000, Oracle, and many other data sources for which there is an OLE DB or ODBC provider. To a certain extent, ADO.NET represents the latest evolution of ADO technology.

2) LINQ to SQL

LINQ to SQL allows .NET developers to write "queries" in their .NET language of choice to retrieve and manipulate data from a SQL Server database. In a general sense, LINQ to SQL allows us to create SQL queries in our preferred .NET language syntax and work with a strongly types collection of objects as a return result. We can make changes to these objects then save changes back to the database.

1. Nhibernate

Nhibernate is a port of Hibernate Core for Java to the .NET Framework. It handles persisting plain .NET objects to and from an underlying relational database. Given an XML description of your entities and relationships, Nhibernate automatically generates SQL for loading and storing the objects. Optionally, you can describe your mapping metadata with attributes in your source code.

Nhibernate supports transparent persistence, your object classes don't have to follow a restrictive programming model.

Nhibernate key features:

- **Natural programming model** – Nhibernate supports natural OO idiom; inheritance, polymorphism, composition and the .NET collections framework, including generic collections.
- **Native .NET** – Nhibernate API uses .NET conventions and idioms
- **Support for fine-grained object models** – a rich variety of mappings for collections and dependent objects
- **No build-time bytecode enhancement** – there's no extra code generation or bytecode processing steps in your build procedure
- **The query options** – Nhibernate addresses both sides of the problem; not only how to get objects into the database, but also how to get them out again
- **Custom SQL** – specify the exact SQL that Nhibernate should use to persist your objects. Stored procedures are supported on Microsoft SQL Server.

- **Support for "conversations"** – Nhibernate supports long-lived persistence contexts, detach/reattach of objects, and takes care of optimistic locking automatically

- **Free/open source** – Nhibernate is licensed under the LGPL (Lesser GNU Public License)

ACCESSING DATA IN JAVA BASED FRAME WORK

There are a variety of open source tools available today for constructing a data access API, which simplify what has been in the past a complicated and error prone mechanism. Before these tools became available, applications resorted to calling JDBC APIs and passing SQL strings to Statement objects to execute data lookup queries. The lookup calls returned ResultSets that an application would use by calling accessor methods matching the data types of the returned columns. While effective, this approach is fragile because it relies on Strings in application code matching the names of database tables and columns; changing the names of database tables or columns required finding all of their references in the code and changing them. This is problematic if not inelegant.

This section will introduce two tools that radically simplify the data access development process as well as a lightweight framework built on top of these tools to hide their implementation details from a client application that wishes to leverage the benefits these tools provide.

There are an abundance of tools and technologies that provide solutions to problems that all data-driven applications share, namely accessing and managing a data model. Two of those tools, Hibernate and Middlegen, combine to automatically generate an object representation of an entity model.

1) **Middlegen:** An open source tool that has the ability to connect to a database server and examine the database metadata to discover table definitions and relationships. As well, Middlegen comes with a plugin for Hibernate that allows it to generate the Hibernate configuration files automatically. The Middlegen and Hibernate code generation utilities can be run from Ant targets they each supply.

2) **Hibernate:** An open-source object/relational mapping toolkit that relieves the need to make direct use of the JDBC API. Hibernate offers facilities for data retrieval and update, transaction management, database connection pooling, programmatic as well as declarative queries, and declarative entity relationship management. Hibernate also has the ability to generate Java source files to match the structure of a database.

XML files containing configuration data provide Hibernate with details about databases with which it needs to interact. These files contain database connection specifics, connection pooling details, transaction factory settings, as well as references to other XML files that describe tables in the database. Combined, these files provide substantial configurability allowing an application to tune the behaviors and performance of its data access layer to a remarkably fine level of granularity.

1.4 SECURING THE DATA

There are numerous ways in which data can be compromised. Below are ways to secure your workstation, E-mail, passwords and internet access.

1.4 Workstation Security

Engaging in good practices with respect to the use of workstations in centers (whether these are in the form of desktop computers or laptops) is crucial. While these devices are highly powerful business tools used for processing and exchanging business information that are capable of enhancing work productivity, if they are deployed insecurely they can provide a point through which the infrastructure of centers can be compromised. Risks that are relevant to workstations include physical theft of or damage to workstations or data assets, infection by viruses or other malware, and unauthorized access and use of confidential information (such as non-public research data or payroll details). This section identifies a number of good practices to facilitate the secure deployment and management of workstations within centers.

- Lock your workstation when you are away from your desk.
- Shut down the workstation each night. (If you are not supported by the Student Services Zone, contact your technical support to see if this applies to you.)
- Make sure that personal or sensitive data about employees, students, customers, or anyone otherwise affiliated with Purdue is not stored on the workstation hard drive, laptops, tablet, PCs, CDs, floppy disks, BlackBerry, or other external devices such as pin drives or any other media subject to confiscation, infiltration or compromise. Personal or sensitive data includes but are not limited to SSN, credit card, and other identification information.
- Store data protected as defined by FERPA, GLBA, HIPAA on departmental servers and not on personal workstations. In addition, storage on servers helps to ensure the integrity of the data with normal backup procedures.
- Empty your Recycle Bin daily.

1.4.2 Passwords

A **password** is a secret word or string of characters that is used for authentication, to prove identity or gain access to a resource (example: an access code is a type of password). The password should be kept secret from those not allowed access.

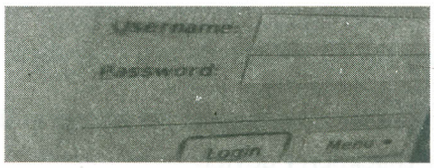


Fig. 1: Data authentication using password

The use of passwords is known to be ancient. Sentries would challenge those wishing to enter an area or approaching it to supply a password or *watchword.* Sentries would only allow a person or group to pass if they knew the password. In modern times, user names and passwords are commonly used by people during a log in process that controls access to protected computer operating systems, mobile phones, cable TV decoders, automated teller machines (ATMs), etc. A typical computer user may require passwords for many purposes: logging in to computer accounts, retrieving e-mail from servers, accessing programs, databases, networks, web sites, and even reading the morning newspaper online.

Despite the name, there is no need for passwords to be actual words; indeed passwords which are not actual words may be harder to guess, a desirable property. Some passwords are formed from multiple words and may more accurately be called a passphrase. The term **pass code** is sometimes used when the secret information is purely numeric, such as the personal identification number (PIN) commonly used for ATM access. Passwords are generally short enough to be easily memorized and typed.

For the purposes of more compellingly authenticating the identity of one computing device to another, passwords have significant disadvantages (they may be stolen, spoofed, forgotten, etc.) over authentications systems relying on cryptographic protocols, which are more difficult to circumvent.

Good password has the following qualities:

- It is at least seven characters long
- It is easy enough for you to remember that you do not need to write it down
- It includes both upper and lower case letters
- It includes digits and/or punctuation characters as well as letters
- It does not use proper names, such as, Washington, Harry, Bob, etc.

- It does not use personal information, such as, your phone number, street address, pet's name, etc..
- It is not a dictionary searchable word in any language.

By numerous ways you can secure your password, some of them are as follows:

- Always use strong passwords and keep them secret.
- Do not log in for other people for access to the computer system, e-mail system or Blackberry device.
- Do not save passwords (mainframe, ftp, website passwords, etc.) to your workstation hard drive, E-mail or blackberry.

1.4.3 E-Mail Security

E-mail messages are composed, delivered, and stored in a multiple step process, which starts with the message's composition. When the user finishes composing the message and sends it, the message is transformed into a standard format: an RFC 2822 formatted message. Afterwards, the message can be transmitted. Using a network connection, the mail client, referred to as a mail user agent (MUA), connects to a mail transfer agent (MTA) operating on the mail server. The mail client then provides the sender's identity to the server. Next, using the mail server commands, the client sends the recipient list to the mail server. The client then supplies the message. Once the mail server receives and processes the message, several events occur: recipient server identification, connection establishment, and message transmission. Using Domain Name System (DNS) services, the sender's mail server determines the mail server(s) for the recipient(s). Then, the server opens up a connection(s) to the recipient mail server(s) and sends the message employing a process similar to that used by the originating client, delivering the message to the recipient(s).

E-mail is a security-attack pipeline into your company. E-mail carries essential messages for the everyday workings of your business/organization. But it's also a major conduit for security threats that blend web, E-mail, and data attack strategies. According to the research conducted following facts have been obtained:

- Nearly 90% of unwanted E-mails contain web links to spam or malicious websites.
- Only 1 of every 4 anti-virus products catch blended web and E-mail attack campaigns.
- Cybercriminals use news, shopping, and other hot topics to disguise spam-carrying E-mails so well that even savvy recipients can't resist clicking.

Fig. 2: Electronic mail security

Unified content security is essential for E-mail protection. Companies that don't have unified content solutions that include web, E-mail, and data are sitting targets for costly security attacks.

E-mail is a data-loss pipeline out of your company. One of the top challenges your company faces is loss of confidential data such as client names, employee SSNs, database passwords, and more. If you don't have tight outbound protection, you can be sure that you're losing data right now, due to:

- Accidental misuse of data by employees.
- E-mail links to unprotected databases.
- Nonmalicious but damaging actions such as documenting passwords in E-mail messages or attachments.

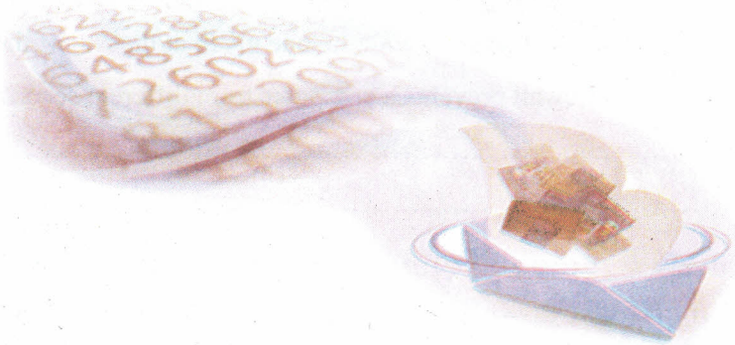


Fig. 3: Risk against confidential data

Leak-free out bound E-mail security is essential for DLP. You must have E-mail security that's tightly integrated into your data loss prevention (DLP) technology. If you don't, you're losing data, inviting crime, risking compliance violations, and jeopardizing your reputation.

The following checklist can be applied to improve your E-mail security:

- Check your e-mail "Sent Items" and "Deleted Items" daily for sensitive data.
- Do not open E-mail attachments that you aren't expecting. Especially avoid attachments ending in .exe, .vbs, .pif, .scr, .com, or .bat, and don't

unzip files you are not expecting. Don't open the attachment even if it looks like it is sent from someone you know as many viruses can forge, or spoof, the sender's name from names found in address books.

- Never store sensitive personal information such as your bank account information or Social Security numbers on your hard drive of your computer, your e-mail account, or Blackberry.
- Do not E-mail restricted data. Note: Refer to Student Restricted Data document. The preferred method of delivery for restricted data is hand delivery.

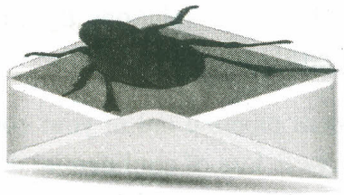


Fig. 4: Threats from E-mail attachments

- Never comply with requests for personal information from an e-mail or phone call unless you initiated the contact. These are often scams trying to steal personal information.

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Explain how firewall prevents a computer from different security problems.

.....

.....

.....

.....

2) Why data security is important?

.....

.....

.....

.....

3) What are the characteristics of a good password? Mention few of them.

.....

.....

.....

.....

4) What is a SMTP connection?

.....

.....

.....

.....

1.4.4 Internet Security

Internet security is a branch of computer security specifically related to the Internet. Its objective is to establish rules and measures to use against attacks over the Internet. The Internet represents an insecure channel for exchanging information leading to a high risk of intrusion or fraud, such as phishing. Different methods have been used to protect the transfer of data, including encryption. Some of the points to keep in mind for internet security are:

- Do not download software such as screensavers, games, or other programs from unfamiliar or unverified sources. These can harbor computer viruses or open a "back door," giving others access to your computer.
- Delete temporary internet files.
- Turn off auto-complete. It stores information such as usernames and passwords.

Table 1: Online threats

The numbers: Internet threats to your securities are real.
▪ In 2002, more than 10 million people were victims of identity theft, costing the average victim more than $1,000 and a year's time to repair their credit.1 ▪ More than 95% of Internet users have inadequate protection from online threats.2 ▪ Over 90% of computer users have dangerous "spyware" lurking on their computers without their knowledge.3 ▪ In 2002, nearly 20 million people had the skills to hack a computer.4 ▪ In 2003, Internet-related identity theft more than tripled.5 ▪ Today, a typical online PC is "scanned" by outside intruders twelve times every day.6 1 – Federal Trade Commission Report 09/03/03 2 – SummitWatch 11/03 3 – TechNewsWorld 3/19/04 4 – Information Warfare Task Force

5 – USA Today 10/23/03

6 – National Strategy to Secure Cyberspace 9/02

1.4.5 Application Security

Application security encompasses measures taken throughout the application's life-cycle to prevent exceptions in the security policy of an application or the underlying system (vulnerabilities) through flaws in the design, development, deployment, upgrade, or maintenance of the application.

Applications only control the use of resources granted to them, and not which resources are granted to them. They, in turn, determine the use of these resources by users of the application through application security.

Table 2: Application Threats/Attacks

Category	Threats / Attacks
Input Validation	Buffer overflow; cross-site scripting; SQL injection; canonicalization
Authentication	Network eavesdropping ; Brute force attack; dictionary attacks; cookie replay; credential theft
Authorization	Elevation of privilege; disclosure of confidential data; data tampering; luring attacks
Configuration management	Unauthorized access to administration interfaces; unauthorized access to configuration stores; retrieval of clear text configuration data; lack of individual accountability; over-privileged process and service accounts
Sensitive information	Access sensitive data in storage; network eavesdropping; data tampering
Session management	Session hijacking; session replay; man in the middle
Cryptography	Poor key generation or key management; weak or custom encryption
Parameter manipulation	Query string manipulation; form field manipulation; cookie manipulation; HTTP header manipulation
Exception management	Information disclosure; denial of service
Auditing and logging	User denies performing an operation; attacker exploits an application without trace; attacker covers his or her tracks

For application security you can keep in mind that social Security Numbers are extremely sensitive information and as they are classified as "restricted" data, written permission must be needed to have access to SSN.

1.4.6 Physical Security

Physical security is the protection of personnel, hardware, programs, networks, and data from physical circumstances and events that could cause serious losses or damage to an enterprise, agency, or institution. This includes protection from fire, natural disasters, burglary, theft, vandalism, and terrorism.

Physical security is often overlooked (and its importance underestimated) in favor of more technical and dramatic issues such as hacking, viruses, Trojans, and spyware. However, breaches of physical security can be carried out with little or no technical knowledge on the part of an attacker. Moreover, accidents and natural disasters are a part of everyday life, and in the long term, are inevitable.

There are three main components to physical security. First, obstacles can be placed in the way of potential attackers and sites can be hardened against accidents and environmental disasters. Such measures can include multiple locks, fencing, walls, fireproof safes, and water sprinklers. Second, surveillance and notification systems can be put in place, such as lighting, heat sensors, smoke detectors, intrusion detectors, alarms, and cameras. Third, methods can be implemented to apprehend attackers (preferably before any damage has been done) and to recover quickly from accidents, fires, or natural disasters.

1.4.7 Taking Backup

Backing Up Your Database With the Oracle-Suggested Disk Backup Strategy also helps in making your data more secure.

Follow these steps to use the Oracle-suggested strategy to back up to disk:

1. In the Backup/Recovery section of the Maintenance page, click Schedule Backup.

   The Schedule Backup page appears.

2. In the Oracle-Suggested Backup section, click the Schedule Oracle-Suggested Backup button.

   The Schedule Oracle-Suggested Backup: Destination page appears. On this page, you select the destination media for the backup, which can be disk, tape or both.

3. Select Disk as the destination and click Next.

   The Schedule Oracle-Suggested Backup: Setup page. This page describes the backups that are performed each day as part of the disk-based strategy.

4. There are no settings to change on this page. Click Next

   The Schedule Oracle-Suggested Backup: Schedule page appears.

5. You are prompted for Start Date, Time Zone, and Daily Backup Time for the daily backups. Based upon your expected usage patterns, choose times for the nightly backup during which database activity is low. Click Next.

   The Schedule Oracle-Suggested Backup: Review page appears.

6. The backup script RMAN will run is displayed (although you cannot edit the script directly). You are presented with a chance to confirm or alter your settings. In the backup script, you can see the tag ORA$OEM_LEVEL_0 that the script assigns to the backup. Assuming you do not need to change the schedule, click Submit Job to add the job for the Oracle-suggested strategy to your schedule.

   Your database will now be backed up once daily, using incremental backups and incrementally applied backups, allowing quick recovery to any time in the preceding 24 hours.

1.4.8 File-level Security

Windows XP supports both FAT32 and NTFS. If you are concerned about security, NTFS is the way to go. For example, it allows you to set permissions at the file level instead of just at the folder level. When you installed the operating system, you may have opted to (or mistakenly) used FAT32. Not a problem since you has a one-time conversion from FAT to NTFS, without losing any of your data!

One way of converting to NTFS is to use the convert command from the command prompt. The syntax for the command is as follows: Convert d: /fs:ntfs

After you press Enter, you'll be asked to confirm your actions by pressing Y and the conversion is done. You can now set security at the file level. Now if your drive is currently in use, prime example is if you are trying to convert your system volume, you can opt to have the conversion take place the next time the computer is restarted.

A word of caution though as this is a one time conversion which means there isn't any going back from NTFS to FAT32 unless you format the volume or find a third party utility that can perform this task.

1.4.9 Disk Encryption

Disk encryption uses disk encryption software or hardware to encrypt every bit of data that goes on a disk or disk volume. Disk encryption prevents unauthorized access to data storage. The term "full disk encryption" (or whole disk encryption) is often used to signify that everything on a disk is encrypted, including the programs that can encrypt bootable operating system partitions. But they must still leave the master boot record (MBR), and thus part of the

disk, unencrypted. There are, however, hardware-based full disk encryption systems that can truly encrypt the entire boot disk, including the MBR.

1.4.10 Public Key Infrastructure

A PKI (public key infrastructure) enables users of a basically unsecure public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority. The public key infrastructure provides for a digital certificate that can identify an individual or an organization and directory services that can store and, when necessary, revoke the certificates. Although the components of a PKI are generally understood, a number of different vendor approaches and services are emerging. Meanwhile, an Internet standard for PKI is being worked on.

The public key infrastructure assumes the use of public key cryptography, which is the most common method on the Internet for authenticating a message sender or encrypting a message. Traditional cryptography has usually involved the creation and sharing of a secret key for the encryption and decryption of messages. This secret or private key system has the significant flaw that if the key is discovered or intercepted by someone else, messages can easily be decrypted. For this reason, public key cryptography and the public key infrastructure is the preferred approach on the Internet. (The private key system is sometimes known as symmetric cryptography and the public key system as asymmetric cryptography.)

A public key infrastructure consists of:

- A certificate authority (CA) that issues and verifies digital certificate. A certificate includes the public key or information about the public key
- A registration authority (RA) that acts as the verifier for the certificate
- authority before a digital certificate is issued to a requestor
- One or more directories where the certificates (with their public keys) are held
- A certificate management system

1.4.11 Using Steganography

Steganography is beneficial for securely storing sensitive data, such as hiding system passwords or keys within other files. However, it can also pose serious problems because it's difficult to detect. Network surveillance and monitoring systems will not flag messages or files that contain steganographic data. Therefore, if someone attempted to steal confidential data, they could conceal it within another file and send it in an innocent looking email.

1.4.12 IP Security

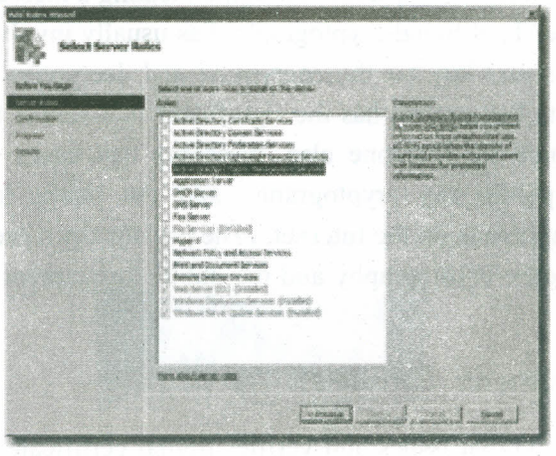
IP security allows individual users or organizations to secure traffic for all applications, without having to make any modifications to the applications. Therefore, the transmission of any data, such as e-mail or application-specific company data can be made secure.

1.4.13 Using Rights Management Services

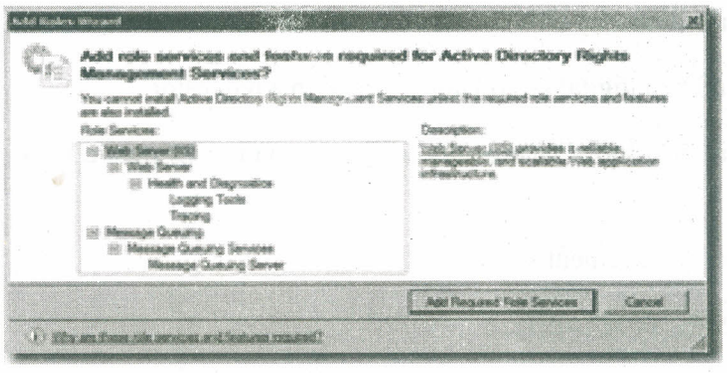
A lot of data that flows around nowadays can be quite sensitive. So how do you protect sensitive data from being forwarded to people who you don't actually want to read it. The solution is to use Active Directory Rights Management and this section shows you the steps to went through to get it to work.

Active Directory Rights Management

To do this you add the role to the server

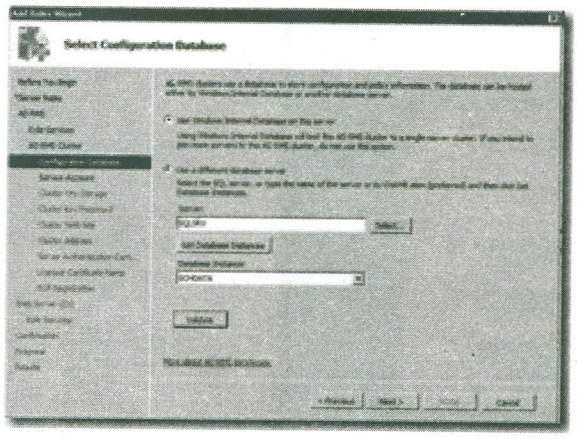


Add the required Services



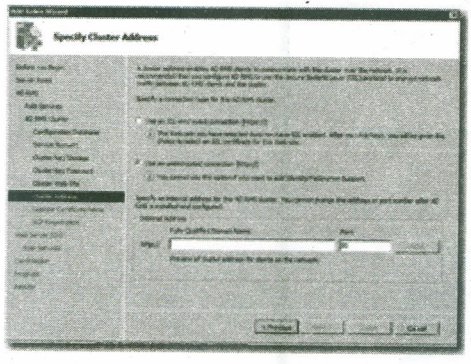
Go through the next screens accepting the defaults until you reach the database screen

Here you can either select to use the Windows internal database to store all the data or select an existing SQL Server. In this case the screenshot shows our SQL 2008 R2 cluster



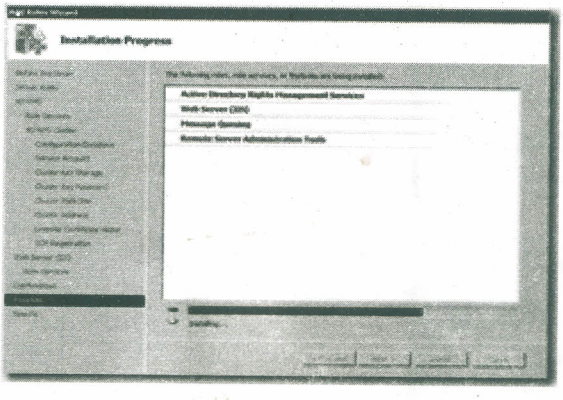
Select a user that has network access to the server and then work through the next screens accepting the defaults until you get to the cluster address screen

At this screen you need to the FQDN of the server and also decide if you are going to use https or just plain old http



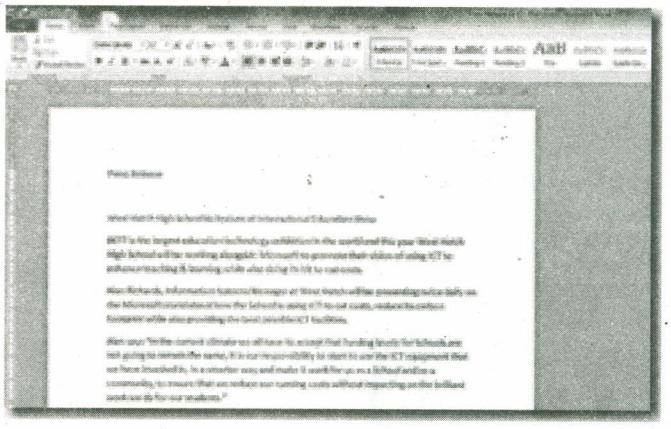
At the next screens accept the defaults and give a name to the certificate store you will remember

The role should now install



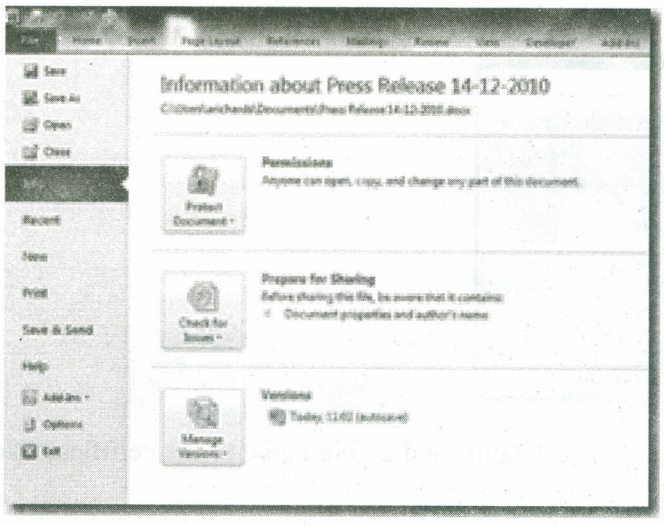
Using Rights Management In Office 2010

Here we have a document in Word 2010 that I only want one other person to be able to read

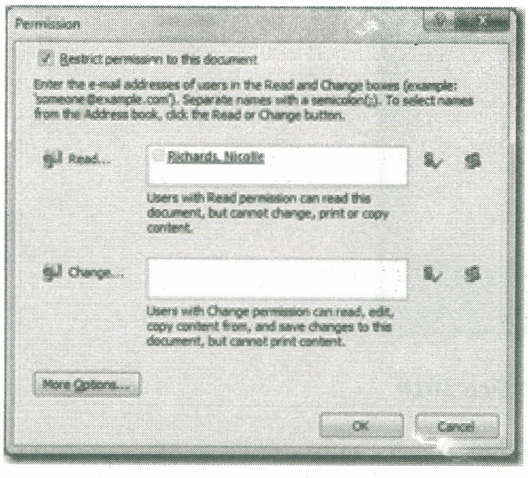


To protect the file click on the **File** tab and from the info option select

Permissions | Protect Document | Restrict Permission By People | Restricted Access



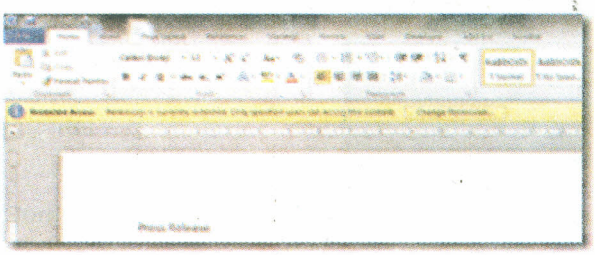
Tick the box to restrict permission and then enter the name of the person you want to be able to access the document



You can set it up so that certain people can read the document and other people can read and change the document

Click **OK** and save the document

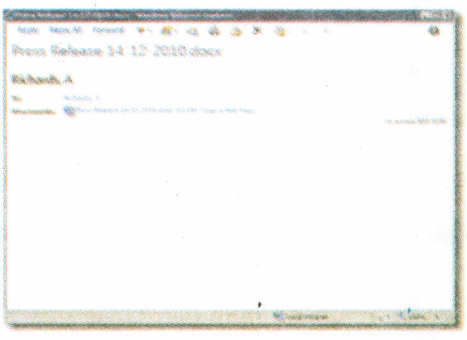
You will now see a yellow bar above the document telling you that you have security in place



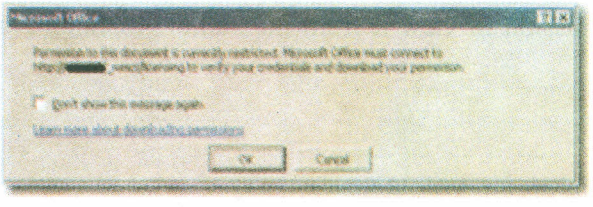
The End User

So what happens when you send the file to the user.

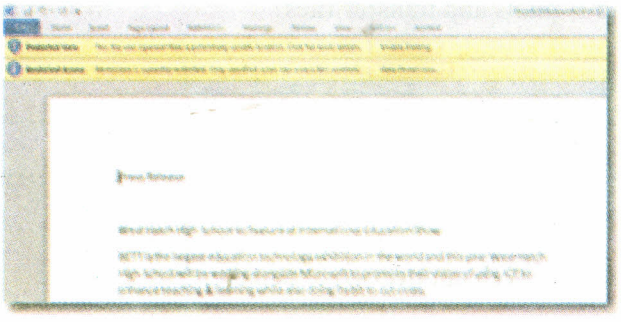
Here is the email containing the file



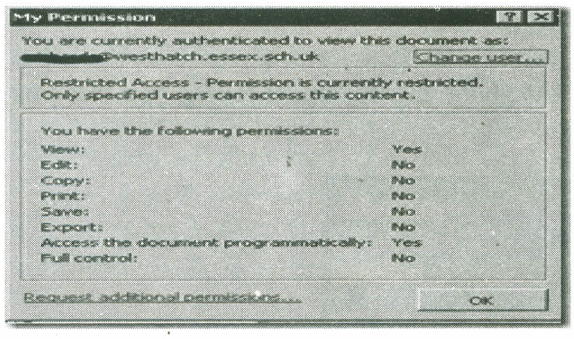
Opening the file brings up the following dialogue box



Once the credentials are downloaded the file opens

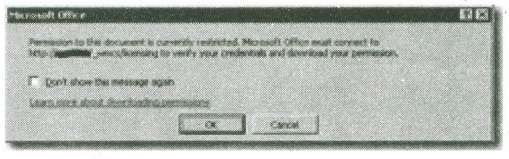


Even though the user can click to enable editing, when they do nothing happens. They can also view the permissions they have by clicking on the **View Permission** button

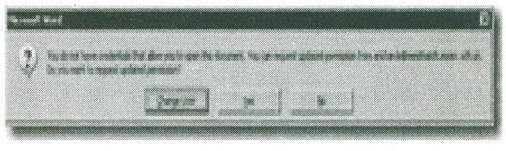


But what happens when it gets forwarded onto a user who does not have permission to view the file.

Open the file and again the first thing that happens is that the user is told that the file has restricted access and it will need to contact the rights management server to download credentials



Once that is done the user is told that they don't have permission



As you can see they can change the user that is being used to access the document or request permission to view it from the author. Clicking **No** will close the document

Check Your Progress 3

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) How to securely send E-mails and transfer files?

.....

.....

.....

.....

2) Name some most common web application attacks/threats.

.....

.....

.....

.....

3) What is the difference between network security and cryptography?

.....

.....

.....

.....

4) What is Network eavesdropping?

.....

.....

.....

.....

5) Why you use Network security?

.....

.....

.....

.....

6) What is Encryption and how is it used with internet?

.....

.....

.....

.....

1.5 LET US SUM UP

This unit is an effort towards answering some of the fundamental queries about Data Access and Internet security. There are various kinds of security controls-access, flow, cryptographic etc and all of them complement each other to provide the security to the important data. Management and reporting capabilities are critical in any data security solution deployment. Not only must they provide simple intuitive interfaces but they must also consolidate many tasks, sometimes spanning multiple security solutions mentioned in this unit. All the preventive measures discussed here to subject to practical limitations which keep them away from achieving their objectives under all conditions. There is no such mechanism which is perfectly secure and provides 100% security against unauthorized data access but they are good enough to reduce the risk of compromise to an acceptable level.

1.6 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Many people still feel insecure about making purchases or sharing information over the Internet. If proper protocols are followed, this is an unreasonable concern. Many other ways of sharing information are less secure than a computer network. Information sent through the mail or a courier service passes through many waypoints on its journey, each point or person offering opportunities for theft or damage. Making purchases over a telephone can be risky because information can be picked up with a radio scanner or in some cases with another telephone. Digital data encryption creates a secure network that allows information to be sent and received without many concerns.

2) There's no question about it: Securing data has changed. Security buzzwords like firewalls, passwords and intrusion detection are now making room for up-and-coming terms like compliance and regulations. It's no longer enough to protect systems from external threats only. And with new laws and regulations, the cost of maintaining system compliance and passing audits continues to grow. Major standards and requirements such as Payment Card Industry (PCI), Sarbanes-Oxley and HIPAA are driving many companies to reconfigure how their data is secured.

3) Threat – Anything that can exploit vulnerability, intentionally or accidentally, and obtain, damage, or destroy an asset.

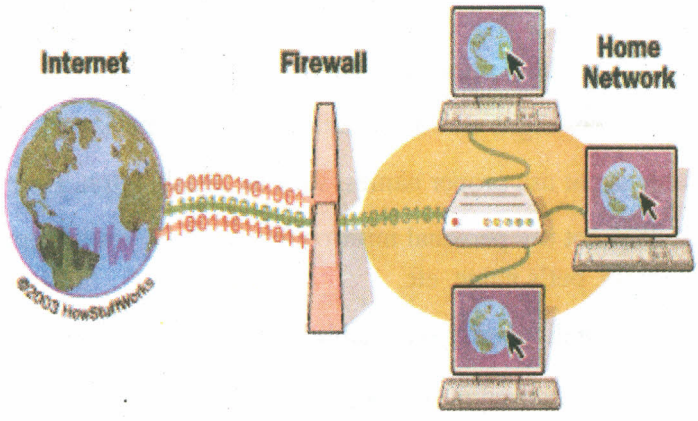
   Vulnerability – Weaknesses or gaps in a security program that can be exploited by threats to gain unauthorized access to an asset.

   Risk – The potential for loss, damage or destruction of an asset as a result of a threat exploiting vulnerability.

4) Spyware is considered as more dangerous than virus:

Virus	Spyware
▪ Damages data	▪ Steals sensitive private information.
▪ Written by hackers	▪ Written by professional online criminals.
▪ Infection is obvious and can be detected with	▪ Infection is silent and cannot by detected anti-virus software with anti-virus software.
▪ Most computer users are sufficiently protected.	▪ Very few computer users are protected
▪ The threat is decreasing	▪ The threat is increasing

1) Firewall is software that works on some set of rules and instructions given by you. A firewall helps to keep your computer more secure and protect from many security problems like; hacking, Trojan horse, Virus, etc. It restricts information that comes to your computer from other computers, giving you more control over the data on your computer and providing a line of defense against people or programs (including viruses and worms) that try to connect to your computer without invitation.



Working of a Firewall

2) **Data security** is important to most businesses. Financial information such as accounts and tax details, or employee information - including payroll and personnel records - could be very difficult to replace. This could expose you to certain risks that need managing carefully. If you lost data through human error, fire, and theft or for some other reason, you would at the very least have to spend time and effort collecting and reproducing the information.

More seriously, loss of confidential or sensitive data could expose you to the risk of fraud or copyright breaches. Your sales, distribution and the reputation of your business could be directly affected. Projects in progress - e.g. new product designs - could be delayed as the work is redone.

Losing data in a customer database - such as customer names, contact details and information on their buying habits - could prevent you targeting customers with appropriate mail shots or informing them of new products. This could mean you lose potential sales, and revenue.

A **virus** can damage your business by making documents stored on computers unusable. As more and more business is conducted via email, a virus can also make getting in contact with suppliers and customers more difficult. This can mean delays in making purchase orders and taking customer orders.

You should also remember that you have a responsibility to safeguard any personal data that you store. You may be committing an offence under the Data Protection Act if you don't guard against unauthorized access to, and accidental loss of, damage to or destruction of personal data.

3) A good password has the following qualities:

   - It is at least seven characters long
   - It is easy enough for you to remember that you do not need to write it down
   - It includes both upper and lower case letters
   - It includes digits and/or punctuation characters as well as letters
   - It does not use proper names, such as, Washington, Harry, Bob, etc.
   - It does not use personal information, such as, your phone number, street address, pet's name, etc.
   - It is not a dictionary searchable word in any language.

4) **SMTP** is an acronym for Simple Mail Transfer Protocol.SMTP is an acronym for Simple Mail Transfer Protocol. It is an Internet standard outlined by Internet Engineering Task Force (IETF) to be used for sending email messages. All internet providers nowadays use this protocol to send email. When using SMTP, there should be a client sending the messages and the server receiving them. Your email program utilizes this protocol and acts as SMTP client to distribute email messages to recipients. In most cases you will configure and use any email client the same way you do with your regular email client like Microsoft Outlook. You will specify the SMTP server that Internet provider gave you. When the SMTP Client sends email messages, it connects to the SMTP server you have specified and communicates to it using SMTP protocol.

Check Your Progress 3

1) Even in this day and age, after decades of having the Internet, we are still surprisingly sending and receiving most of our e-mails and information insecurely through the World Wide Web. We're still using most of the founding protocols and technologies that transfer data in what we call clear-text. When in clear-text, your passwords and the data content can be captured on the local network you're connected to, which is an even bigger problem when connected to a public Wi-Fi hotspot or Internet port. Additionally, the data could be captured and read by hackers or eavesdroppers during stops at servers when it's moving through the Web.

   There are two main concerns when speaking about e-mail security--the security of the link between the e-mail server and e-mail client, or whether or not it's encrypted; and the security of the e-mail message content and

any attachments, or whether or not it's encrypted during transmission and storage. These two concerns apply to each the sender and recipient(s).To address the first concern, you can use a Web-based client or e-mail service where you can login through a Web browser. Just make sure you're logging on through an HTTPS address. You should see a padlock in the lower right side of your browser or next to the address bar on top. If you must use an e-mail client program (such as Outlook or Thunderbird), try to use SSL encryption, which your e-mail provider must support. To address the second concern, you could encrypt your e-mail messages. The traditional method is to use PGP encryption with digital certificates to get a private and public encryption key. You'd send people your public key so they could send you encrypted messages that you'd decrypt with your private key.

An alternative method to address the second, or message encryption, issue you might find easier is to use an online secure messaging service. Of course, this depends on how well you trust the service provider. Send, for example, lets you quickly send and receive secure messages and attachments without creating a new e-mail address. If you'd like a new e-mail provider, consider those that offer security features, such as Hush mail.

2) Some of the most common web applications attacks are:

- Anonymous Proxy Vulnerabilities
- Brute Force Login
- Buffer Overflow
- Cookie Injection
- Cookie Poisoning
- Corporate Espionage
- Credit Card Exposure
- Cross Site Request Forgery (CSRF)
- Known Worms
- Malicious Encoding

3) **Cryptography** is the deliberate attempt to obscure or scramble the information so that only an authorized receiver can see the message. Network security may employ cryptography, but has many other tools to secure a network, including firewalls, auditing, Intrusion Detection Systems, and so forth.

Cryptography would be used only when trying to keep messages secret when sending them across a network or keeping information secret in a file.

4) **Network Eavesdropping** or network sniffing is a network layer attack consisting of capturing packets from the network transmitted by others' computers and reading the data content in search of sensitive information like passwords, session tokens, or any kind of confidential information. The attack could be done using tools called network sniffers. These tools collect packets on the network and, depending on the quality of the tool, analyze the collected data like protocol decoders or stream reassembling. Depending on the network context, for the sniffing to be the effective, some conditions must be met:

• LAN environment with HUBs

This is the ideal case because the hub is a network repeater that duplicates every network frame received to all ports, so the attack is very simple to implement because no other condition must be met.

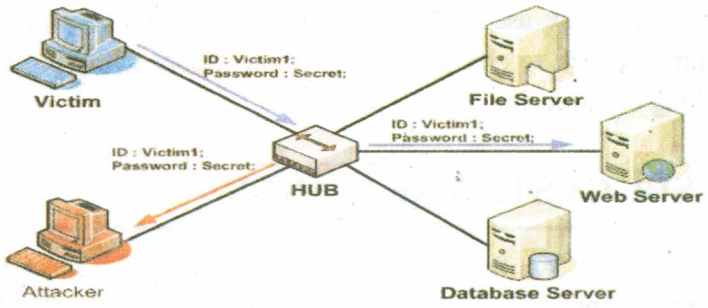
• LAN environment with switches

To be effective for eavesdropping, a preliminary condition must be met. Because a switch by default only transmits a frame to the port, a mechanism that will duplicate or will redirect the network packets to an evil system is necessary. For example, to duplicate traffic from one port to another port, a special configuration on the switch is necessary. To redirect the traffic from one port to another, there must be a preliminary exploitation like the arp spoof attack. In this attack, the evil system acts like a router between the victim's communications, making it possible to sniff the exchanged packets.

• WAN environment

In this case, to make a network sniff it's necessary that the evil system becomes a router between the client server communications. One way to implement this exploit is with a DNS spoof attack to the client system.

Network Eavesdropping is a passive attack which is very difficult to discover. It could be identified by the effect of the preliminary condition or, in some cases, by inducing the evil system to respond a fake request directed to the evil system IP but with the MAC address of a different system. Consider an example, when a network device called a HUB is used on the Local Area Network topology, the Network Eavesdropping become easier because the device repeats all traffic received on one port to all other ports. Using a protocol analyzer, the attacker can capture all traffic on the LAN discovering sensitive information.



Local Eavesdropping attack

5) It's used to protect your information while you are communicating with bank servers and so on. It's already established that information can be stolen during transfers; network security helps to prevent such things.

6) Since the internet has become such an important tool to access an endless source of information, having a system to securely perform this on your computer is extremely important to a large majority of people today.

   A process called Encryption becomes vital for a secure and safe environment for the computers and the internet. The basis of encryption comes from the age old process called cryptography, which is basically secret writing that needs to be decoded. Since cryptography is a code that relies on human input, computers are needed to have a secure process by which the code can be cracked and a secure transmission is delivered. Encryption is a method by which any kind of messages such as email, e-commerce, banking, or any other personal information, become encoded so they are illegible to anyone other the intended person without a special deciphering of the code, with a special key or other code. This form of security is necessary to handle the privacy of almost all personal and private information that passes through from one computer to another. Information such as data or messages that are sent is regarded as plain text until that information is encrypted and then is labeled as cipher text.

   Here is a list of several ways encryption is used on the internet today:

   - Banks and other financial institutions
   - Credit card companies
   - Private correspondence (e-mail)
   - Information for applications, for example, social security numbers, individual personal profile data, and any other personal profile information.
   - Company profiles and private information

Computers may use either one of two encryption schemes; public key or symmetric key, to convert plain text into cipher text with the end result of locking or unlocking data.

1.7 SUGGESTED READINGS

- Aktan, B., C.A. Bohus, L.A. Crowl and M.H. Shor(1996). Distance Learning Applied to Control Engineering Laboratories. IEEE Transactions on Education, 39(3), 320-326.
- David Salomon, Data Privacy and Security, Springer.
- Henry, J. (1996). Controls Laboratory Teaching via the World Wide Web. In: Proceedings of the ASEE Annual Conference. Washington, USA, Session 3513 Hughes Technologies (1999). Mini SQL (mSQL).
- http://www.Hughes.com.au/products/msql/The Center for Imaginary Environments (1999).mSQL-JDBC.
- http://www.imaginary.com/Java/Soul/Jochheim, A. and C. R¨ohrig (1999). The Virtual Lab for Teleoperated Control of Real Experiments.In: Proceedings of the 38th IEEE Conference on Decision and Control. Phoenix, USA, 1, 819-824.
- Ondrejkov´a, R. (1998). Graphical User Interface to Evaluate Experiments in Virtual Lab. Diploma Thesis, University of Hagen, Slovak Technical University Bratislava.Overstreet, J. W. and A. Tzes (1999). An Internet-Based Real-Time Control Engineering Laboratory, IEEE Control Systems Magazine, 19(5), 19-34.
- R. Perlman, Network Security: Private Communication in a Public World - 2nd ed.", Prentice Hall 2002 (ISBN 0130460192).
- William Stallings and Lawrence Brown, Computer Security: Principles and Practice, Prentice Hall.

UNIT 2 ERROR HANDLING AND LOGGING

Structure

2.0 INTRODUCTION

All the sectors of economy are required by legal and regulatory requirements to be:

- Auditable – all activities that affect user state or balances are formally tracked
- Traceable – it's possible to determine where an activity occurs in all tiers of the application
- High integrity – logs cannot be overwritten or tampered with by local or remote users

Well-written applications will dual-purpose logs and activity traces for audit and monitoring, and make it easy to track a transaction without excessive effort or access to the system. They should possess the ability to easily track or identify potential fraud or anomalies end-to-end.

Error handling and logging are different aspects of the same topic: how to track events within an application. It is the best step to adapt some good strategies like: Fail safe – do not fail open, Dual purpose logs, Audit logs are legally protected – protect them and Reports and search logs using a read-only copy or complete replica.

2.1 OBJECTIVES

After studying this unit, you should be able to describe:

- Error Handling;
- Error Reporting and Error Suppression; and
- Logging.

2.2 ERROR HANDLING

Error handling is an important part of any real-world application. This provides a number of mechanisms that can be used to handle errors, both during the development process and once your application is in a production environment.

Error handling takes two forms: structured exception handling and functional error checking. Structured exception handling is always preferred as it is easier to cover 100% of code. Functional languages, such as PHP 4, that do not have exceptions are very hard to cover 100% of all errors. Code that covers 100% of errors is extraordinarily verbose and difficult to read, and can contain subtle bugs and errors in the error handling code itself. Motivated attackers like to see

error messages as they might leak information that leads to further attacks, or may leak privacy related information. Web application error handling is rarely robust enough to survive a penetration test. Applications should always fail safe. If an application fails to an unknown state, it is likely that an attacker may be able to exploit this indeterminate state to access unauthorized functionality, or worse create, modify or destroy data.

2.2.1 Fail Safe

- Inspect the application's fatal error handler.
- Does it fail safe? If so, how?
- Is the fatal error handler called frequently enough?
- What happens to in-flight transactions and ephemeral data?

2.2.2 Debug Errors

- Does production code contain debug error handlers or messages?
- If the language is a scripting language without effective pre-processing or compilation, can the debug flag be turned on in the browser?
- Do the debug messages leak privacy related information, or information that may lead to further successful attack?

2.2.3 Exception Handling

- Does the code use structured exception handlers (try {} catch {} etc) or function-based error handling?
- If the code uses function-based error handling, does it check every return value and handle the error appropriately?
- Would fuzz injection against the average interface fail?

2.2.4 Functional Return Values

Many languages indicate an error condition by return value. E.g.:

```
$query = mysql_query("SELECT * FROM table WHERE id=4", $conn);

if ($query === false) {// error}
```

- Are all functional errors checked? If not, what can go wrong?

2.3 ERROR REPORTING

Normally, when an error occurs in a script, the error message is inserted into the script's output. If the error is fatal, the script execution stops.

There are three levels of conditions: notices, warnings, and errors. A notice is a condition encountered while executing a script that could be an error but could also be encountered during normal execution (e.g., trying to access a variable that has not been set). A warning indicates a nonfatal error condition; typically, warnings are displayed when calling a function with invalid arguments. Scripts will continue executing after issuing a warning. An error indicates a fatal condition from which the script cannot recover. A parse error is a specific kind of error that occurs when a script is syntactically incorrect. All errors except parse errors are runtime errors.

By default, all conditions except runtime notices are caught and displayed to the user. User can change this behavior globally in your php.ini file with the error_reporting option. User can also locally change the error-reporting behavior in a script using the error_reporting( ) function.

With both the error_reporting option and the error_reporting( ) function, you specify the conditions that are caught and displayed by using the various bitwise operators to combine different constant value.

2.4 ERROR SUPPRESSION

User can disable error messages for a single expression by putting the error suppression operator @ before the expression. For example:

```
$value = @(2 / 0);
```

Without the error suppression operator, the expression would normally halt execution of the script with a "divide by zero" error. As shown here, the expression does nothing. The error suppression operator cannot trap parse errors, only the various types of runtime errors.

To turn off error reporting entirely, use:

```
error_reporting(0);
```

This ensures that, regardless of the errors encountered while processing and executing your script, no errors will be sent to the client (except parse errors, which cannot be suppressed). Of course, it doesn't stop those errors from occurring.

2.5 TRIGGERING ERRORS

User can throw an error from within a script with the trigger_error( ) function:

```
trigger_error(message [, type]);
```

The first parameter is the error message; the second, optional, parameter is the

condition level, which is either E_USER_ERROR, E_USER_WARNING, or E_USER_NOTICE (the default).

Triggering errors is useful when writing your own functions for checking the sanity of parameters.

2.6 DEFINING ERROR HANDLERS

Error handling refers to the anticipation, detection, and resolution of programming, application, and communications errors. Specialized programs, called error handlers, are available for some applications. The best programs of this type forestall errors if possible, recover from them when they occur without terminating the application, or (if all else fails) gracefully terminate an affected application and save the error information to a log file.

In programming, a development error is one that can be prevented. Such an error can occur in syntax or logic. Syntax errors, which are typographical mistakes or improper use of special characters, are handled by rigorous proofreading. Logic errors, also called bugs, occur when executed code does not produce the expected or desired result. Logic errors are best handled by meticulous program debugging. This can be an ongoing process that involves, in addition to the traditional debugging routine, beta testing prior to official release and customer feedback after official release.

A run-time error takes place during the execution of a program, and usually happens because of adverse system parameters or invalid input data. An example is the lack of sufficient memory to run an application or a memory conflict with another program. On the Internet, run-time errors can result from electrical noise, various forms of malware or an exceptionally heavy demand on a server. Run-time errors can be resolved, or their impact minimized, by the use of error handler programs, by vigilance on the part of network and server administrators, and by reasonable security countermeasures on the part of Internet users.

If there is a need to better error control than just hiding any errors (and you usually do), user needs to supply PHP with an error handler. The error handler is called when a condition of any kind is encountered and can do anything you want it to, from logging to a file to pretty-printing the error message. The basic process is to create an error-handling function and register it with set_error_handler( ).

The function declare can take in either two or five parameters. The first two parameters are the error code and a string describing the error. The final three parameters, if your function accepts them, are the filename in which the error occurred, the line number at which the error occurred, and a copy of the active symbol table at the time the error happened. Your error handler should check

the current level of errors being reported with error_reporting( ) and act appropriately.

The call to set_error_handler( ) returns the current error handler. You can restore the previous error handler either by calling set_error_handler( ) with the returned value when your script is done with its own error handler, or by calling the restore_error_handler( ) function.

Example: SQL Runtime Error Handling

In SQL, an error condition is called an exception. Exceptions can be internally defined (by the runtime system) or user defined. Examples of internally defined exceptions include division by zero and out of memory. Some common internal exceptions have predefined names,

such as ZERO_DIVIDE and STORAGE_ERROR. The other internal exceptions can be given names.

Define exceptions of your own in the declarative part of any SQL block, subprogram, or package. For example, you might define an exception named insufficient_ funds to flag overdrawn bank accounts. Unlike internal exceptions, user-defined exceptions must be given names.

When an error occurs, an exception is raised. That is, normal execution stops and control transfers to the exception-handling part of your PL/SQL block or subprogram. Internal exceptions are raised implicitly (automatically) by the run-time system. User-defined exceptions must be raised explicitly by RAISE statements, which can also raise predefined exceptions.

To handle raised exceptions, you write separate routines called exception handlers. After an exception handler runs, the current block stops executing and the enclosing block resumes with the next statement. If there is no enclosing block, control returns to the host environment.

Example calculates a price-to-earnings ratio for a company. If the company has zero earnings, the division operation raises the predefined exception ZERO_DIVIDE, the execution of the block is interrupted, and control is transferred to the exception handlers. The optional OTHERS handler catches all exceptions that the block does not name specifically.

Example 1 Runtime Error Handling

```
DECLARE
  stock_price NUMBER := 9.73;
  net_earnings NUMBER := 0;
  pe_ratio NUMBER;
```

```
BEGIN
-- Calculation might cause division-by-zero error.
  pe_ratio := stock_price / net_earnings;
  DBMS_OUTPUT.PUT_LINE('Price/earnings ratio = ' || pe_ratio);
EXCEPTION  -- exception handlers begin
-- Only one of the WHEN blocks is executed.
  WHEN ZERO_DIVIDE THEN  -- handles 'division by zero' error
    DBMS_OUTPUT.PUT_LINE('Company must have had zero earnings.');
    pe_ratio := NULL;
  WHEN OTHERS THEN  -- handles all other errors
    DBMS_OUTPUT.PUT_LINE('Some other kind of error occurred.');
    pe_ratio := NULL;
END;  -- exception handlers and block end here
/
```

The example illustrates exception handling. With some better error checking, we could have avoided the exception entirely, by substituting a null for the answer if the denominator was zero, as shown in the following example.

```
DECLARE
  stock_price NUMBER := 9.73;
  net_earnings NUMBER := 0;
  pe_ratio NUMBER;
BEGIN
  pe_ratio :=
    CASE net_earnings
      WHEN 0 THEN NULL
      ELSE stock_price / net_earnings
    end;
END;
```

2.7 DETAILED ERROR MESSAGES

Detailed error messages provide attackers with a mountain of useful information.

2.7.1 How to Determine If You Are Vulnerable?

- Are detailed error messages turned on?
- Do the detailed error messages leak information that may be used to stage a further attack, or leak privacy related information?
- Does the browser cache the error message?

2.7.2 How to Protect Yourself?

Ensure that your application has a "safe mode" to which it can return if something truly unexpected occurs. If all else fails, log the user out and close the browser window.

Production code should not be capable of producing debug messages. If it does, debug mode should be triggered by editing a file or configuration option on the server. In particular, debug should not enabled be an option in the application itself.

If the framework or language has a structured exception handler (i.e. try {} catch {}), it should be used in preference to functional error handling.

If the application uses functional error handling, its use must be comprehensive and thorough.

Detailed error messages, such as stack traces or leaking privacy related information, should never be presented to the user. Instead a generic error message should be used. This includes HTTP status response codes (i.e. 404 or 500 Internal Server error).

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) What are the two forms of error handling?

.....

.....

.....

.....

.....

2) Which code should not be capable of producing debug messages?

.....

.....

.....

.....

.....

3) In which mode your application to which it can return if something unexpected occurs?

.....

.....

.....

.....

.....

2.8 LOGGING

2.8.1 Where to Log To?

Logs should be written so that the log file attributes are such that only new information can be written (older records cannot be rewritten or deleted). For added security, logs should also be written to a write once / read many devices such as a CD-R.

Copies of log files should be made at regular intervals depending on volume and size (daily, weekly, monthly, etc.). A common naming convention should be adopted with regards to logs, making them easier to index. Verification that logging is still actively working is overlooked surprisingly often, and can be accomplished via a simple job.

Make sure data is not overwritten.

Log files should be copied and moved to permanent storage and incorporated into the organization's overall backup strategy.

Log files and media should be deleted and disposed of properly and incorporated into an organization's shredding or secure media disposal plan. Reports should be generated on a regular basis, including error reporting and anomaly detection trending.

Be sure to keep logs safe and confidential even when backed up.

2.8.2 Handling

Logs can be fed into real time intrusion detection and performance and system monitoring tools. All logging components should be synced with a timeserver so that all logging can be consolidated effectively without latency errors. This time server should be hardened and should not provide any other services to the network.

No manipulation, no deletion while analyzing.

2.8.3 General Debugging

Logs are useful in reconstructing events after a problem has occurred, security related or not. Event reconstruction can allow a security administrator to determine the full extent of an intruder's activities and expedite the recovery process.

2.8.4 Forensics Evidence

Logs may in some cases be needed in legal proceedings to prove wrongdoing. In this case, the actual handling of the log data is crucial.

2.8.5 Attack Detection

Logs are often the only record that suspicious behavior is taking place: Therefore logs can sometimes be fed real-time directly into intrusion detection systems.

2.8.6 Quality of Service

Repetitive polls can be protocol led so that network outages or server shutdowns get protocolled and the behavior can either be analyzed later on or a responsible person can take immediate actions.

2.8.7 Proof of Validity

Application developers sometimes write logs to prove to customers that their applications are behaving as expected.

- Required by law or corporate policies.
- Logs can provide individual accountability in the web application system universe by tracking a user's actions.

It can be corporate policy or local law to be required to (for example) save header information of all application transactions. These logs must then be kept safe and confidential for six months before they can be deleted.

The points from above show all different motivations and result in different requirements and strategies. This means, that before we can implement a

logging mechanism into an application or system, we have to know the requirements and their later usage. If we fail in doing so this can lead to unintentional results.

Failure to enable or design the proper event logging mechanisms in the web application may undermine an organization's ability to detect unauthorized access attempts, and the extent to which these attempts may or may not have succeeded. We will look into the most common attack methods, design and implementation errors, as well as the mitigation strategies later on in this chapter.

There is another reason why the logging mechanism must be planned before implementation. In some countries, laws define what kind of personal information is allowed to be not only logged but also analyzed. For example, in Switzerland, companies are not allowed to log personal information of their employees (like what they do on the internet or what they write in their emails). So if a company wants to log a worker's surfing habits, the corporation needs to inform her of their plans in advance.

This leads to the requirement of having anonymized logs or de-personalized logs with the ability to re-personalized them later on if need be. If an unauthorized person has access to (legally) personalized logs, the corporation is acting unlawful. So there can be a few (not only) legal traps that must be kept in mind.

2.8.8 Logging Types

Logs can contain different kinds of data. The selection of the data used is normally affected by the motivation leading to the logging. This section contains information about the different types of logging information and the reasons why we could want to log them.

In general, the logging features include appropriate debugging information such as time of event, initiating process or owner of process, and a detailed description of the event. The following are types of system events that can be logged in an application. It depends on the particular application or system and the needs to decide which of these will be used in the logs:

- Reading of data file access and what kind of data is read. This not only allows to see if data was read but also by whom and when.
- Writing of data logs also where and with what mode (append, replace) data was written. This can be used to see if data was overwritten or if a program is writing at all.
- Modification of any data characteristics, including access control permissions or labels, location in database or file system, or data

ownership. Administrators can detect if their configurations were changed.

- Administrative functions and changes in configuration regardless of overlap (account management actions, viewing any user's data, enabling or disabling logging, etc.)
- Miscellaneous debugging information that can be enabled or disabled on the fly.
- All authorization attempts (include time) like success/failure, resource or function being authorized, and the user requesting authorization. We can detect password guessing with these logs. These kinds of logs can be fed into an Intrusion Detection system that will detect anomalies.
- Deletion of any data (object). Sometimes applications are required to have some sort of versioning in which the deletion process can be cancelled.
- Network communications (bind, connect, accept, etc.). With this information an Intrusion Detection system can detect port scanning and brute force attacks.
- All authentication events (logging in, logging out, failed logins, etc.) that allow to detect brute force and guessing attacks too.

2.9 DIFFERENT TYPES OF SECURITY OBSTRUCTIONS

2.9.1 Noise

Noise is intentionally invoking security errors to fill an error log with entries (noise) that hide the incriminating evidence of a successful intrusion. When the administrator or log parser application reviews the logs, there is every chance that they will summarize the volume of log entries as a denial of service attempt rather than identifying the 'needle in the haystack'.

2.9.1.1 How to Protect Yourself?

This is difficult since applications usually offer an unimpeded route to functions capable of generating log events. If you can deploy an intelligent device or application component that can shun an attacker after repeated attempts, then that would be beneficial. Failing that, an error log audit tool that can reduce the bulk of the noise, based on repetition of events or originating from the same source for example. It is also useful if the log viewer can display the events in order of severity level, rather than just time based.

2.9.2 Cover Tracks

In logging mechanism attacks goes to the contender who can delete or manipulate log entries at a granular level, "as though the event never even happened!" Intrusion and deployment of rootkits allows an attacker to utilize specialized tools that may assist or automate the manipulation of known log files. In most cases, log files may only be manipulated by users with root / administrator privileges, or via approved log manipulation applications. As a general rule, logging mechanisms should aim to prevent manipulation at a granular level since an attacker can hide their tracks for a considerable length of time without being detected. Simple question; if you were being compromised by an attacker, would the intrusion be more obvious if your log file was abnormally large or small, or if it appeared like every other day's log?

2.9.2.1 How to Protect Yourself?

Assign log files the highest security protection, providing reassurance that you always have an effective 'black box' recorder if things go wrong. This includes:

- Applications should not run with Administrator, or root-level privileges. This is the main cause of log file manipulation success since super users typically have full file system access. Assume the worst case scenario and suppose your application is exploited. Would there be any other security layers in place to prevent the application's user privileges from manipulating the log file to cover tracks?
- Ensuring that access privileges protecting the log files are restrictive, reducing the majority of operations against the log file to alter and read.
- Ensuring that log files are assigned object names that are not obvious and stored in a safe location of the file system.
- Writing log files using publicly or formally scrutinized techniques in an attempt to reduce the risk associated with reverse engineering or log file manipulation.
- Writing log files to read-only media (where event log integrity is of critical importance).
- Use of hashing technology to create digital fingerprints. The idea is that if an attacker does manipulate the log file, then the digital fingerprint will not match and an alert generated.
- Use of host-based IDS technology where normal behavioral patterns can be 'set in stone'. Attempts by attackers to update the log file through anything but the normal approved flow would generate an exception and the intrusion can be detected and blocked. This is one security

control that can safeguard against simplistic administrator attempts at modifications.

2.9.3 False Alarms

Taking cue from the classic 1966 film "How to Steal a Million", or similarly the fable of Aesop; "The Boy Who Cried Wolf", be wary of repeated false alarms, since this may represent an attacker's actions in trying to fool the security administrator into thinking that the technology is faulty and not to be trusted until it can be fixed.

2.9.3.1 How to Protect Yourself?

Simply be aware of this type of attack, take every security violation seriously, always get to the bottom of the cause event log errors rather, and don't just dismiss errors unless you can be completely sure that you know it to be a technical problem.

2.9.4 Denial of Service

By repeatedly hitting an application with requests that cause log entries, multiply this by ten thousand and the result is that you have a large log file and a possible headache for the security administrator. Where log files are configured with a fixed allocation size, then once full, all logging will stop and an attacker has effectively denied service to your logging mechanism. Worse still, if there is no maximum log file size, then an attacker has the ability to completely fill the hard drive partition and potentially deny service to the entire system. This is becoming more of a rarity though with the increasing size of today's hard disks.

2.9.4.1 How to Protect Yourself?

The main defense against this type of attack are to increase the maximum log file size to a value that is unlikely to be reached, place the log file on a separate partition to that of the operating system or other critical applications and best of all, try to deploy some kind of system monitoring application that can set a threshold against your log file size and/or activity and issue an alert if an attack of this nature is underway.

2.9.5 Destruction

Following the same scenario as the Denial of Service above, if a log file is configured to cycle round overwriting old entries when full, then an attacker has the potential to do the evil deed and then set a log generation script into action in an attempt to eventually overwrite the incriminating log entries, thus destroying them.

If all else fails, then an attacker may simply choose to cover their tracks by purging all log file entries, assuming they have the privileges to perform such actions. This attack would most likely involve calling the log file management program and issuing the command to clear the log, or it may be easier to simply delete the object which is receiving log event updates (in most cases, this object will be locked by the application). This type of attack does make an intrusion obvious assuming that log files are being regularly monitored, and does have a tendency to cause panic as system administrators and managers realize they have nothing upon which to base an investigation on.

2.9.5.1 How to Protect Yourself?

Following most of the techniques suggested above will provide good protection against this attack. Keep in mind two things:

- Administrative users of the system should be well trained in log file management and review. 'Ad-hoc' clearing of log files is never advised and an archive should always be taken. Too many times a log file is cleared, perhaps to assist in a technical problem, erasing the history of events for possible future investigative purposes.
- An empty security log does not necessarily mean that you should pick up the phone and fly the forensics team in. In some cases, security logging is not turned on by default and it is up to you to make sure that it is. Also, make sure it is logging at the right level of detail and benchmark the errors against an established baseline in order measure what is considered 'normal' activity.

2.9.6 Audit Trails

Audit trails are legally protected in many countries, and should be logged into high integrity destinations to prevent casual and motivated tampering and destruction.

2.9.6.1 How to Determine If You Are Vulnerable?

- Do the logs transit in the clear between the logging host and the destination?
- Do the logs have similar tamper proofing mechanism to prevent change from the time of the logging activity to when it is reviewed?
- Can relevant logs be easily extracted in a legally sound fashion to assist with prosecutions?

2.9.6.2 How to Protect Yourself?

- Only audit truly important events – you have to keep audit trails for a long time, and debug or informational messages are wasteful

- Log centrally as appropriate and ensure primary audit trails are not kept on vulnerable systems, particularly front end web servers
- Only review copies of the logs, not the actual logs themselves
- Ensure that audit logs are sent to trusted systems
- For highly protected systems, use write-once media or similar to provide trust worthy long term log repositories
- For highly protected systems, ensure there is end-to-end trust in the logging mechanism. World writeable logs, logging agents without credentials (such as SNMP traps, syslog etc.) are legally vulnerable to being excluded from prosecution

2.10 PROBLEM AND SOLUTION-SCALABLE FROM SMALL TO LARGE SYSTEMS IN ERROR HANDLING AND LOGGING

In the process of tackling the problem several well-known patterns are used. Error handling is universal to small and large systems, and whatever mechanism user uses to handle errors there is always a need to log errors. Error handling and logging is one of the key reasons why small solutions often don't scale upwards.

E.g.

In a 500-line program user can send error reports to standard error, or a message box. In a 50,000-line program that is distributed across multiple machines and compiled from re-usable libraries something more substantial is required.

The solution is designed to scale from small systems to large systems.

The problem

Three principal requirements are:

- The error log should be easily accessible and easy to use: developers should be encouraged to log early, log often.
- It should be possible to log to multiple destinations (sinks.)
- It should be possible to add and remove sinks at run time.

In a console application user traditionally send error messages to the console on stderr/cerr, and in a GUI user typically send the errors to message boxes for the user to read and dismiss. In larger systems log errors to a file, database,

remote error logging component or an OS based log, e.g. the NT Event log and UNIX syslog.

There is a need to log errors to different places at different times, although it is hard to beat a simple text file for audit purposes. In more exotic configurations it may be desirable to dynamically change logging sinks: e.g. when connected to the Internet send error reports to the home site.

To build there will be a need of:

- One chain of responsibility design pattern: each destination sink is listed in a map, the message is passed to each sink which has the opportunity to do something with it.
- Two singleton design patterns: all messages for logging are channeled to a single point, from here they are distributed to the various sinks.
- One very small strategy pattern: this is used to reduce the amount of code needed when iterating over the map.
- One proxy pattern: using a proxy output stream allows messages to be streamed to the log.
- One namespace: suitable for implementing a sub-system.
- One critical section: this allows the model to be used in multi-threaded systems.
- Classification of messages system: here divided into Error messages, Warning messages (potential errors), Information messages (not an error, but something worth noting) and Debug messages (something the developer thought might be useful to know.)
- Pointers to members: not a typo but a lesser know aspect of C++
- Initialization of non-local objects

Error Logging is a Sub-System

- Error handling should be built at all levels of a system from the bottom up. Hence, any logging mechanism should inhabit one of the lowest layers. This layer exhibits several characteristics:
- High integrity code: the code will not be executed often (we hope) but when it executes we expect it to work perfectly, the last thing we want is an error in the error logging. Paradoxically, the most important code is some of the least used. As usual the solution is to keep the code short and simple.
- Good performance and limited risks : if the system is failing we may

not have much time left, further we don't know what has gone wrong, it would be better to avoid allocating memory (we may be out of memory), performing GUI actions (the GUI may be stalled), and so on. Of cause each system will be different. When there is an error to report have to do it fast and at minimum risk.

- The log should be accessible from any point in the program or libraries: an error may occur at any point, if the logging mechanism is wrapped in class we must have a pointer or reference to the object before we can log. Either make this a global or pass it as a parameter.
- The sub-system should be stable: so built it into our libraries.
- The system should allow for expansion: if corporate policy mandates that all logs should be made to a central server we should be able to add this with minimum source code changes.

Overview of Patterns

Singleton

This is probably the easiest pattern to understand and one of the most widely used. The pattern name is an accurate summary of the pattern: the is only one of this thing. Usually, the thing is a class, so a singleton pattern typically occurs where there is only one instance of a particular class.

In this example, there may be many log sinks, but there is only one point of logging. Behind this is a singleton object, the SinkStore that implements the functionality, it would be pointless to have more than one SinkStore object. Because SinkStore is hidden behind a namespace I have not enforced the singleton pattern with program code.

The simplest way to ensure you have a singleton in your system is to add a static member to the class. Initialise it to zero and have the class constructor check it is zero before incrementing it. If the member is ever non-zero in the constructor this cannot be the only object of this class so throw and excepetion or assert out.

Proxy

A second singleton exists in this system. The LogStream class which acts to as a proxy, it provides access to the log functions via the stream out operator. Often a proxy is used to restrict access to a function or object, but in this instance we are using a proxy to provide syntactic sugar, making it easier to log messages.

A proxy is best summed up as: "a stand-in for the real image." Distributed technologies (DCOM and CORBA) frequently use proxies on a local machine

to hide the location of the real target. A proxy may do very little, as in this case, or it may perform some work (e.g. marshalling parameters for network transfer in DCOM), but it never performs the actual task you requested.

Chain of Responsibility

In this pattern a chain of handlers - here implemented with a map but any traversable structure may be used - is formed. A message, or other object, is passed to each handler in turn, allowing each to do something with it. Each handler implements functionality without the object being aware of what is being done. Hence we decouple the object being acted upon from the action performed, adding actions becomes easy.

In this implementation each message is passed to each link in the chain. One variation - which is used in Microsoft ATL for windows messages - is to have each handle return a "handled flag." When a handler returns true the chain is terminated.

Strategy

In a strategy we know what we want to do, we can define an interface to represent what we want to do, but we don't want to get our hands dirty with the actual details. We are concerned with the bigger picture. In this case we are concerned with ensuring that the message is logged, we don't care about how or where.

A strategy allows us to vary the algorithm we are using while keeping the big picture the same. The standard C++ library containers are good examples of this. We know how to iterate over them, and iterating over each one is the same, but how each container does it is different from container to container, and even implementation to implementation.

Strategy relies on encapsulating the interface to a process in such a way that the process can differ without effecting the general technique.

Design

The sub-system presented here replies on a singleton logger. This is implemented as set of functions in a namespace. As a namespace we can provide global access to the logging functions without the dangers present in global variables.

The logger namespace presents an interface that hides an implementation using a map. When a message is sent for logging it is passed to every handler in the map allowing it to be handled any log sink.

All elements in the map must be derived from the abstract base class LogSink. This provides the interface that allows the chain and strategy pattern to work.

```
class LogSink
{
public:
 explicit LogSink() {}
 virtual ~LogSink() = 0 {}
 virtual void Debug(conststd::stringand) const = 0;
 virtual void Information(conststd::stringand) const = 0;
 virtual void Warning(conststd::stringand) const = 0;
 virtual void Error(conststd::stringand) const = 0;
};
```

Elements of the map are created on the heap by the main program and adopted by the sub-system. Once adopted the main program can choose to forget about them, they will be deleted when the application is closed. If however, we decide to remove one of the elements we may request it to be *orphaned*, at which time is removed from the map and returned to the caller for deletion. To make this possible all elements are named.

Messages may be sent to the logger either by direct function calls: Debug, Information, Warning and Error; or by way of a stream class which may accept elements of any type which operate with the standard streams. (You may add additional overloads to provide syntactic sugar for other string types you may be using.) A single output channel is provided - accessed through MsgOut() - for sending these classes to the logger.

```
namespace Log {
 void AdoptSink(conststd::stringand name, LogSink* const);
LogSink* OrphanSink(conststd::stringand name);
 void Debug(conststd::stringand);
 void Information(conststd::stringandmsg);
 void Warning(conststd::stringandmsg);
 void Error(conststd::stringandmsg);
}
```

Multi-threaded systems are allowed for by guarding critical sections. This code should present a minimal burden on single-threaded systems.

Strategy for Chain Handlers

When a message is received it is passed to each handler in turn. Each element is free to implement an empty function if it wishes, or take whatever action it considers most appropriate.

The actual iteration is performed by the WriteMsg function regardless of the actual method being called; the function implements the iteration, the actual mechanics, are irrelevant, or to put it another way, the strategy is to call a method on each handler. This is implemented using a pointer to a member.

As it's name suggests, a pointer to a member is not a function call itself, but is a pointer, to a function call. Unlike pointers to functions it is not an absolute memory address but an offset into the v-table of an object, it also ensures the object called is available. It is perfect for this application where all the objects are derived from a single base class, and all the functions which may be called are similar virtual (indeed pure-virtual) functions of the base class. Importantly the member functions are interchangeable.

What is actually present is a very small strategy pattern: the members form "a family of algorithms" which are "interchangeable" which "lets the algorithm vary independently from... use." In future a "new operation" can be defined "without changing the classes on which is performed."

Handler Elements

Handlers must be derived from LogSink class to ensure a common interface. They may be placed in the Log namespace if you wish but this is not essential. They are added to the chain using the AdoptSink function and removed with the OrphanSink function. When a handler is removed the responsibility for deleting it passes to the callee.

Three example handlers:

standard error handler: writes to stderr/cerr

file error handler: writes to a file you name

box error handler: which puts a Windows message box on screen, while being Win32 specific this shows how different message types can be handled differently.

SinkStore

SinkStore class encapsulates the list of handler. By implementing this as a class and using the static local trick we are sure it is initialised before first use, and that the destructor is called at close to free any resources. LogSink objects owned by the store will also get a chance to free resources (e.g. file handles, data base handles) when their destructors are called.

SinkStore only exists in the implementation of the log sub-system, i.e. the Log.cpp file. Therefore we are free to change the implementation without incuring any side effects, or even the need to recompile dependent code. Given that the log sub-system is in the bottom layer and permeates all aspects of the system this should reduce the need to rebuild all the source code after a tweak.

Mixing Patterns

Patterns are often used in combination, e.g. abstract factories to produce singletons; visitors implementing strategies; proxies and just about anything. Where one pattern starts and ends can sometimes be difficult to see. In truth, it doesn't matter where one starts and ends. The program will still compile if you cross two patterns. A pattern presents a way of looking at a problem. At least one of the patterns in this example was discovered after the code was written.

2.11 ACTION TO BE TAKEN IN ERROR HANDLING AND LOGGING

All applications have failures – whether they occur during compilation or runtime. Most programming languages will throw runtime exceptions for illegally executing code (e.g. syntax errors) often in the form of cryptic system messages. These failures and resulting system messages can lead to several security risks if not handled properly including; enumeration, buffer attacks, sensitive information disclosure, etc. If an attack occurs it is important that forensics personnel be able to trace the attacker's tracks via adequate logging.

Error Handling

Hackers can use the information exposed by error messages. Even missing templates errors (HTTP 404) can expose your server to attacks (e.g. buffer overflow, XSS, etc.). If you enable the Robust Exception Information debugging option, ColdFusion will display:

Physical path of template

URI of template

Line number and line snippet

SQL statement used (if any)

Data source name (if any)

Java stack trace

ColdFusion provides tags and functions for developers to use to customize error handling. Administrators can specify default templates in the ColdFusion

Administrator (CFAM) to handle unknown or unhandled exceptions. ColdFusion's structure exception handling works in the following order:

Template level (ColdFusion templates and components)

ColdFusion exception handling tags: cftry, cfcatch, cfthrow, and cfrethrow

try and catch statements in CFScript

Application level (Application.cfc/cfm)

Specify custom templates for individual exceptions types with the cferror tag

Application.cfconError method to handle uncaught application exceptions

System level (ColdFusion Administrator settings)

Missing Template Handler execute when a requested ColdFusion template is not found

Site-wide Error Handler executes globally for all unhandled exceptions on the server

Actions to be taken

- Do not allow exceptions to go unhandled
- Do not allow any exceptions to reach the browser
- Display custom error pages to users with an email link for feedback
- Do not enable "Robust Exception Information" in production.
- Specify custom pages for ColdFusion to display in each of the following cases:
    - When a ColdFusion page is missing (the Missing Template Handler page)
    - When an otherwise-unhandled exception error occurs during the processing of a page (the Site-wide Error Handler page)
    - You specify these pages on the Settings page in the Server Settings are in the ColdFusion MX Administrator; for more information, see the ColdFusion MX Administrator Help.
- Use the cferror tag to specify ColdFusion pages to handle specific types of errors.
- Use the cftry, cfcatch, cfthrow, and cfrethrow tags to catch and handle exception errors directly on the page where they occur.

- In CFScript, use the try and catch statements to handle exceptions.
- Use the onError event in Application.cfc to handle exception errors that are not handled by try/catch code on the application pages.

Logging

Log files can help with application debugging and provide audit trails for attack detection. ColdFusion provides several logs for different server functions. It leverages the log libraries for customized logging. It also provides logging tags to assist in application debugging.

The following is a partial list of ColdFusion log files and their descriptions

Log file	Description
application.log	Records every ColdFusion MX error reported to a user. Application page errors, including ColdFusion MX syntax, ODBC, and SQL errors, are written to this log file.
exception.log	Records stack traces for exceptions that occur in ColdFusion.
scheduler.log	Records scheduled events that have been submitted for execution. Indicates whether task submission was initiated and whether it succeeded. Provides the scheduled page URL, the date and time executed, and a task ID.
server.log	Records start up messages and errors for ColdFusion MX.
customtag.log	Records errors generated in custom tag processing.
mail.log	Records errors generated by an SMTP mail server.
mailsent.log	Records messages sent by ColdFusion MX.
flash.log	Records entries for Macromedia Flash Remoting.

It contains the Logging Settings and log viewer screens. Administrators can configure the log directory, maximum log file size, and maximum number of archives. It also allows administrators to log slow running pages, CORBA calls, and scheduled task execution. The log viewer allows viewing, filtering, and searching of any log files in the log directory (default is cf_root/logs). Administrators can archive, save, and delete log files as well. The cflog and cftrace tags allow developers to create customized logging. <cflog> can write custom messages to the Application.log, Scheduler.log, or a custom log file. The custom log file must be in the default log directory – if it does not exist ColdFusion will create it. <cftrace> tracks execution times, logic flow, and

variable at the time the tag executes. It records the data in the cftrace.log (in the default logs directory) and can display this info either inline or in the debugging output of the current page request. Use <cflog> to write custom error messages, track user logins, and record user activity to a custom log file. Use <cftrace> to track variables and application state within running requests.

Actions to be taken

- Use <cflog> for customized logging
- Incorporate into custom error handling
- Record application specific messages
- Actively monitor and fix errors in ColdFusion's logs
- Optimize logging settings
- Rotate log files to keep them current
- Keep files size manageable
- Enable logging of slow running pages
- Set the time interval lower than the configured Timeout Request value in the CFAM Settings screen
- Long running page timings are recorded in the server log
- Use <cftrace> sparingly for audit trails
- Use with inline="false"
- Use it to track user input – Form and/or URL variables

Check Your Progress 2

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) How does a word Handler derived from?

.....

.....

.....

.....

.....

2) What is the responsibility if the Handler is removed?

.....

.....

.....

.....

.....

3) Where does the SinkStore Exist?

.....

.....

.....

.....

.....

4) How does a pattern use?

.....

.....

.....

.....

.....

5) What happens if an attack occurs in errcr handling and logging?

.....

.....

.....

.....

.....

6) How can Hackers use the information?

.....

.....

.....

.....

.....

2.12 LET US SUM UP

The code written in an extendable fashion so that new handlers can be added at

any time while the core implementation can also be changed without side effects.

Using a namespace subsystem instead of a class removes the need to pass a pointer or reference to all points in the code where an message may be logged. Hence compile time dependencies are reduced and parameter lists are free from clutter.

This sets out to demonstrate how design patterns can help software scale from the smallest to largest systems by looking at a real life problem. The pattern implementations are somewhat smaller but embody the same principals.

2.13 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Structured exception handling and functional error checking.

2) Production Code

3) Safe Mode

Check Your Progress 2

1) Handlers must be derived from LogSink class to ensure a common interface. They may be placed in the Log namespace if you wish but this is not essential. They are added to the chain using the AdoptSink function and removed with the OrphanSink function.

2) When a handler is removed the responsibility for deleting it passes to the callee.

3) SinkStore only exists in the implementation of the log sub-system, i.e. the Log.cpp file.

4) Patterns are often used in combination, e.g. abstract factories to produce singletons; visitors implementing strategies; proxies and just about anything.

5) If an attack occurs it is important that forensics personnel be able to trace the attacker's tracks via adequate logging.

6) Hackers can use the information exposed by error messages.

2.14 SUGGESTED READINGS

- *Bjarne Stroustrup's FAQ*. .research.att.com. 2009-10-04. Retrieved 2011-12-15.

- Cameron, D.; Faust, P.; Lenkov, D. and Mehta, M. *A portable implementation of C++ exception handling*, Proceedings of the C++ Conference (August 1992) USENIX.
- Hutton, Graham and Wright, Joel. *Compiling Exceptions Correctly.* Proceedings of the 7th International Conference on Mathematics of Program Construction, 2004.
- http://www.open-std.org/jtc1/sc22/wg21/docs/papers/1997/N1077.asc
- Jim. *All Exceptions Are Handled.*
- Lajoie, Josée (March–April 1994). *Exception handling – Supporting the runtime mechanism.* C++ Report 6 (3).
- Meyers, Scott (2006). *The Most Important C++ Software...Ever.*
- Wilcox,http://poliTechnosis.kataire.com/2008/02/all-exceptions-are-handled.html

UNIT 3 SERVER CONFIGURATION AND CODE MANAGEMENT

Structure

3.0 INTRODUCTION

An organization's servers provide a wide variety of services to internal and external users, and many servers also store or process sensitive information for the organization. The most common types of servers are Web, email, database, infrastructure management, and file servers.

Servers are frequently targeted by attackers because of their valuable data and services. For example, a server might contain personally identifiable

information that could be used to perform identity theft. Common security threats to servers are:

- Malicious entities may exploit software bugs in the server or its underlying operating system to gain unauthorized access to the server.
- Denial of service (DoS) attacks may be directed to the server or its supporting network infrastructure, denying or hindering valid users from making use of its services.
- Sensitive information on the server may be read by unauthorized individuals or changed in an unauthorized manner.
- Sensitive information transmitted unencrypted or weakly encrypted between the server and the client may be intercepted.
- Malicious entities may gain unauthorized access to resources elsewhere in the organization's network via a successful attack on the server.
- Malicious entities may attack other entities after compromising a server. These attacks can be launched directly (e.g., from the compromised host against an external server) or indirectly (e.g., placing malicious content on the compromised server that attempts to exploit vulnerabilities in the clients of users accessing the server).

The key guidelines for maintaining a secure server:

- Carefully plan and address the security aspects of the deployment of a server.
- Implement appropriate security management practices.
- Ensure that the server operating system is deployed, configured, and managed to meet the security requirements of the organization.
- Ensure that the server application is deployed, configured, and managed to meet the security requirements of the organization.
- Commit to the process of maintaining the security of servers to ensure continued security.

3.1 OBJECTIVES

After studying this unit, you should be able to describe:

- the fundamental activities performed as part of securing and maintaining the security of servers;
- what makes a web server secure;

- how to secure middle tier application servers?
- what a secure database server includes; and
- code review.

3.2 BACKGROUND

3.2.1 Basic Server Security Steps

A number of steps are required to ensure the security of any server. As a prerequisite for taking any step, however, it is essential that the organization have a security policy in place. Implementing the following steps for server security should be effective:

1. Plan the installation and deployment of the operating system (OS) and other components for the server. Section 3 addresses this issue.
2. Install, configure, and secure the underlying OS.
3. Install, configure, and secure the server software.
4. For servers that host content, such as Web servers (Web pages), database servers (databases), and directory servers (directories), ensure that the content is properly secured. This is highly dependent on the type of server and the type of content.
5. Employ appropriate network protection policy (e.g., firewall, packet filtering router, and proxy). Choosing the policies for a particular situation depends on many factors like the location of the server's clients, the location of the server on the network, the types of services offered by the server, and the types of threats against the server.
6. Employ secure administration and maintenance processes, including application of patches and upgrades, monitoring of logs, backups of data and OS, and periodic security testing.

3.2.2 Server Security Principles

While addressing server security issues, the following general information security principles should be kept in mind:

- Simplicity
- Fail-Safe
- Complete Mediation
- Open Design
- Separation of Privilege

- Least Privilege
- Least Common Mechanism
- Psychological Acceptability
- Defense-in-Depth
- Work Factor
- Compromise Recording

3.3 SERVER SECURITY PLANNING

The most critical aspect of deploying a secure server is careful planning before installation, configuration, and deployment. Ensure, through careful planning, that the server is and in compliance with all relevant organizational policies. Many server security and performance problems can be traced to a lack of planning or management controls.

1 Installation and Deployment Planning

2 Security Management Staff

3 Management Practices

4 System Security Plan

5 Human Resources Requirements

3.3.1 Securing the Server Operating System

After planning the installation and deployment of the OS and installing the OS, the following steps are necessary for the security of OS:

- Patch and update the OS
- Harden and configure the OS to address security adequately
- Install and configure additional security controls, if needed
- Test the security of the OS to ensure that the previous steps adequately addressed all security issues

3.3.2 Securing the Server Software

After the OS has been installed and secured, the next step is to install and secure the chosen server software. A partially configured and/or patched server should not be exposed to external networks (e.g., the Internet) or external users. In addition, internal network access should be as limited as possible until all software is installed, patched, and configured securely. Insecure servers can be compromised in a matter of minutes after being placed on the Internet. While it is ideal, but not always feasible, to fully harden the platform before placing it on the network.

3.3.3 Maintaining the Security of the Server

Administrators need to maintain its security continuously after deploying a server.

Primary activities include handling and analyzing log files, performing regular server backups, recovering from server compromises, testing server security regularly, and performing remote administration securely.

3.4 SECURING WEB SERVER

Web server configuration plays a critical role in your Web application's security. Badly configured virtual directories or a common mistake can lead to unauthorized access. A forgotten share can provide a convenient back door, whereas an overlooked port can be an attacker's front door. Neglected user accounts can permit an attacker to slip by your defenses unnoticed.

This section provides a systematic approach that can be use to configure a secure Web server successfully.

3.4.1 Design Guidelines

Consider your Web application's configuration management functionality carefully. Most applications require interfaces that allow content developers, operators, and administrators to configure the application and manage items such as Web page content, user accounts, user profile information, and database connection strings. If remote administration is supported, how are the administration interfaces secured? The consequences of a security breach to an administration interface can be severe, because the attacker frequently ends up running with administrator privileges and has direct access to the entire site.

Web application's configuration management can be improved by practicing the following:

- Secure your administration interfaces.
- Secure your configuration store.
- Maintain separate administration privileges.
- Use least privileged process and service accounts.

3.4.2 Methodology

Many configuration settings can be applied to reduce the server's vulnerability to attack. The best approach is to organize the precautions that must be taken and the settings must be configured into categories. Categorization allows to walk through the securing process from top to bottom systematically or pick a particular category and complete specific steps.

Configuration Categories

The security methodology has been divided into the following categories:

- **Patches and Updates**

  When some new vulnerability is discovered, the code to exploit it is posted on Internet bulletin boards within hours of the first successful attack. If you do not patch and update your server, it provides opportunities for attackers and malicious code. Patching and updating your server software is a first step towards securing your Web server.

- **Services**

  Services are prime vulnerability points for attackers who can exploit the privileges and capabilities of a service to access the local Web server. Do not run a service on your Web server if it is not necessary and if it is necessary, secure it and maintain it.

- **Protocols**

  Avoid the usage of protocols which are inherently insecure. If cannot be avoided, take the appropriate measures to provide secure authentication and communication. Examples of insecure, clear text protocols are Telnet, Post Office Protocol (POP3), Simple Mail Transfer Protocol (SMTP), and File Transfer Protocol (FTP).

- **Accounts**

  Accounts grant authenticated access to the computer, and these must be audited. What is the purpose of the user account? How much access does it have? Is it a common account that can be targeted for attack? Configure accounts with least privilege to help prevent elevation of privilege. Remove the accounts that are not required.

- **Files and Directories**

  Secure all files and directories with restricted permissions that only allow access to necessary Windows services and user accounts. Windows auditing can be used to detect any suspicious or unauthorized activity.

- **Shares**

  Remove all unnecessary file shares as well as the default administration shares if they are not required. Secure the remaining shares with restricted permissions.

- **Ports**

  Services that run on the server listen to specific ports so that they can respond to incoming requests. Audit the ports on your server regularly

to ensure that an insecure or unnecessary service is not active on your Web server. If any active port is detected which was not opened by an administrator then this is a sign of unauthorized access and a security compromise.

- **Registry**

  Many security-related settings are stored in the registry, so the security of the registry is must. This can be done by applying restricted Windows ACLs and by blocking remote registry administration.

- **Auditing and Logging**

  Auditing is one of most important tools for identifying intruders, attacks in progress, and evidence of attacks that have occurred. A combination of Windows and IIS auditing features can be used to configure auditing on the Web server. Event and system logs also help in troubleshooting security problems.

- **Sites and Virtual Directories**

  Sites and virtual directories are directly exposed to the Internet. Even though secure firewall configuration and defensive ISAPI filters such as URLScan (which ships with the IISLockdown tool) can block requests for restricted configuration files or program executables, a more defensive in depth strategy is recommended.

- **Script Mappings**

  Remove all unnecessary IIS script mappings for optional file extensions to prevent an attacker from exploiting any bugs in the ISAPI extensions that handle these types of files. Unused extension mappings are often overlooked and represent major security vulnerability.

- **ISAPI Filters**

  Vulnerabilities in ISAPI filters can be easily attacked. Remove unnecessary ISAPI filters from the Web server.

- **IIS Metabase**

  The IIS metabase maintains IIS configuration settings. The security related settings must be appropriately configured, and the access to the metabase file should be restricted with hardened NTFS permissions.

- **Code Access Security**

  Restrict code access security policy settings to ensure that code downloaded from the -Internet or Intranet have no permissions and will not be allowed to execute.

IIS Installation Recommendations

While installing and configuring a new Web server, follow the procedure given below:

To build a new Web server

1. Install Windows 2000 Server, but do not install IIS as part of the operating system installation.
2. Apply the latest service packs and patches to the operating system.
3. Install IIS separately by using **Add/Remove Programs** in the Control Panel.

Including Service Packs with a Base Installation

If you need to build multiple servers, you can incorporate service packs directly into your Windows installations. Service packs include a program called Update.exe to combine a service pack with your Windows installation files.

3.4.3 Steps

The process of securing your Web server uses the configuration categories. Each step contains one or more actions to secure a particular area or feature.

Step 1: Patches and Updates
Step 2: IIS Lockdown
Step 3: Services
Step 4: Protocols
Step 5: Accounts
Step 6: Files and Directories
Step 7: Shares
Step 8: Ports
Step 9: Registry
Step 10: Auditing and Logging
Step 11: Sites and Virtual Directories
Step 12: Script Mappings
Step 13: ISAPI Filters
Step 14: IIS Metabase
Step 15: Server Certificates
Step 16: Machine.config
Step 17: Code Access Security

Step 1: Patches and Updates

Update your server with the latest service packs and patches. You must update and patch all of the Web server components.

Step 2: IISLockdown

The IISLockdown tool helps you to automate certain security steps. IISLockdown greatly reduces the vulnerability of a Windows 2000 Web server. The custom templates either disable or secure various features. The filter can

be used to block specific HTTP requests to prevent potentially harmful requests from reaching the server and causing damage.

Step 3: Services

Services that do not authenticate clients, services that use insecure protocols, or services that run with too much privilege are risks. If you do not need them, do not run them. If you run a service, make sure that it is secure and maintained. Run the service using a least privilege account, and keep the service updated by applying patches.

Step 4: Protocols

By preventing the use of unnecessary protocols, you reduce the potential for attack. For example, you can control whether your Web Services can use HTTP GET, POST or SOAP.

Step 5: Accounts

You should remove accounts that are not used because an attacker might discover and use them. Use strong passwords. Weak passwords increase the likelihood of a successful brute force or dictionary attack. Use least privilege. An attacker can use accounts with too much privilege to gain access to unauthorized resources.

Step 6: Files and Directories

Install Windows 2000 and Windows Server 2003 on partitions formatted with the NTFS file system so that you benefit from NTFS permissions to restrict access. Use strong access controls to protect sensitive files and directories.

Additional Considerations

Also consider removing unnecessary Data Source Names (DSNs). Only those DSNs required by Web applications should be installed on the Web server.

Step 7: Shares

By default all users have full control on newly created file shares. Harden these default permissions to ensure that only authorized users can access files exposed by the share.

Step 8: Ports

Services that run on the server use specific ports so that they can serve incoming requests. Close all unnecessary ports and perform regular audits to detect new ports in the listening state, which could indicate unauthorized access and a security compromise.

Step 9: Registry

The registry is the repository for many server configuration settings. You must ensure that only authorized administrators have access to it. If an attacker is able to edit the registry, he can reconfigure and compromise the security of your server.

Step 10: Auditing and Logging

Auditing does not prevent system attacks, although it is an important aid in identifying intruders and attacks in progress, and can assist you in diagnosing attack footprints. Enable a minimum level of auditing on your Web server and use NTFS permissions to protect the log files so that an attacker cannot cover his tracks by deleting or updating the log files in any way.

Step 11: Sites and Virtual Directories

Relocate Web roots and virtual directories to a non-system partition to protect against directory traversal attacks. These attacks allow an attacker to execute operating system programs and utilities. It is not possible to traverse across drives.

Step 12: Script Mappings

Script mappings associate a particular file extension, such as .asp, to the ISAPI extension that handles it, such as Asp.dll. IIS is configured to support a range of extensions including .asp, .shtm, .hdc, and so on.

The main security issues associated with script mappings are:

- **An attacker could exploit a vulnerability found in an extension.**

  This could occur if vulnerability in an extension remains unpatched. Unused extensions increase the area of potential attack.

- **Server-side resources could be downloaded by the client.**

  This could occur when a file extension is not mapped correctly. Files that should not be directly accessible by the client should either be mapped to the appropriate handler, based on its extension, or should be removed.

Step 13: ISAPI Filters

Vulnerabilities in ISAPI filters have caused significant IIS exploitation. There are no unneeded ISAPI filters after a clean IIS installation, although the .NET Framework installs the ASP.NET ISAPI filter (Aspnet_filter.dll), which is loaded into the IIS process address space (Inetinfo.exe) and is used to support cookie-less session state management.

Step 14: IIS Metabase

Security and other IIS configuration settings are maintained in the IIS metabase file. Harden the NTFS permissions on the IIS metabase (and the backup metabase file) to be sure that attackers cannot modify your IIS configuration in any way (for example, to disable authentication for a particular virtual directory.)

Step 15: Server Certificates

A valid certificate provides secure authentication so that a client can trust the server it is communicating with and secure communication so that sensitive data remains confidential and tamperproof over the network.

Step 16: Machine.Config

The Machine.config file maintains numerous machine wide settings for the .NET Framework, many of which affect security.

Step 17: Code Access Security

Machine level code access security policy is determined by settings in the Security.config file.

3.4.4 Staying Secure

You need to monitor the security state of your server and update it regularly to help prevent newly discovered vulnerabilities from being exploited. To help keep your server secure:

- Audit group membership.
- Monitor audit logs.
- Stay current with service packs and patches.
- Perform security assessments.
- Use security notification services.

3.5 SECURING APPLICATION SERVER

Middle-tier application servers are most often used to host business logic and data access services. This functionality is usually packaged inside Enterprise Services applications or is exposed to front-end Web servers by using middle-tier Web services. This section addresses each technology separately and shows how to secure your application server in each case.

To secure the application server, you must apply an incremental security configuration after the underlying operating system and Internet Information Services (IIS) Web server (if installed) have been locked down.

3.5.1 Methodology

By securing the communication channels to the application server and preventing any hosts other than authorized Web servers from accessing the application server.

3.5.2 Communication Channel Considerations

Sensitive application data and authentication credentials that are sent to and from the application server should be encrypted to provide privacy and integrity. This mitigates the risk associated with eavesdropping and tampering.

If this threat is negligible— for example, because your application is located in a closed and physically secured network — then you do not need to encrypt the traffic. If network eavesdropping is a concern, then you can use SSL, which provides a secure communication channel at the application layer, or IPSec, which provides a transport-level solution. IPSec encrypts all IP traffic that flows between two servers, while SSL allows each application to choose whether or not to provide an encrypted communication channel.

Enterprise Services

The Enterprise Service channel can be secured in one of two ways:

- RPC Encryption
- IPSec

Web Services

Web services are hosted by ASP.NET and IIS, and the services use the HTTP protocol for communication over the network.

SSL or IPSec can be used to secure the communication channel. Alternatively, encryption can be handled at the application layer by encrypting the message payload or the sensitive parts of the payload.

SQL Server

To secure the channel from the application server to SQL Server, use IPSec or SSL. SSL requires a server certificate to be installed on the database server.

3.5.3 Firewall Considerations

Security infrastructure can include internal firewalls on either side of the application server. This section discusses the ports that are opened on these firewalls to support the functionality of application.

Enterprise Services

If you use middle-tier Enterprise Services, configure an internal firewall that separates the Web server and application server to allow DCOM and RPC traffic.

Fig. 3: Typical Enterprise Services firewall port configuration

Restrict the ports required to support DCOM on the internal firewall in two ways:

- Define port ranges.
- Use static endpoint mapping.

Web Services

Web services communicate using SOAP over HTTP; therefore, only open port 80 on the internal firewall.

SQL Server

If a firewall separates the application server from the database server, then connecting to SQL Server through a firewall requires that you configure the client using the SQL Server Client Network Utility and configure the database server using the Server Network Utility. By default, SQL Server listens on TCP port 1433, although this can be changed. The chosen port must be open at the firewall.

3.6 SECURING DATABASE SERVER

There are many ways to attack a database. External attacks may exploit configuration weaknesses that expose the database server. An insecure Web application may also be used to exploit the database. For example, an application that is granted too much privilege in the database or one that does not validate its input can put your database at risk.

Internal threats should not be overlooked. The database user tricked into running malicious code? Could any malicious code on the network compromise the database?

This section takes a step-by-step approach that shows you how to improve your database server's security.

3.6.1 Methodology

Using categories allows you to systematically walk through the securing process from top to bottom or pick a particular category and apply specific steps.

Configuration Categories

The securing methodology has been organized into the categories shown below:

The configuration categories are based on best practices obtained from field experience, customer validation, and the study of secure deployments.

- **Patches and Updates**

  Many security threats exist because of vulnerabilities in operating systems, services, and applications that are widely published and well known. When new vulnerabilities are discovered, attack code is generally posted on Internet bulletin boards within hours of the first successful attack. Patching and updating your server's software is the first step towards securing your database server. There may be cases where vulnerability exists and no patch is available. In these cases, be aware of the details of the vulnerability to assess the risk of attack and take measures accordingly.

- **Services**

  Services are prime vulnerability points for attackers to exploit the privileges and capabilities of the service to access the server and potentially other computers. Some services are designed to run with privileged accounts. By default, database servers generally do not need all services enabled. Disable unnecessary and unused services.

- **Protocols**

  Limit the range of protocols that client computers can use to connect to the database server and make sure you can secure those protocols.

- **Accounts**

  Restrict the number of Windows accounts accessible from the database server to the necessary set of service and user accounts. Use least privileged accounts with strong passwords in all cases.

- **Files and Directories**

  Use NTFS file system permissions to protect program, database, and log files from unauthorized access. Access control lists (ACLs) in conjunction with Windows auditing detect suspicious or unauthorized activity.

- **Shares**

  Remove all unnecessary file shares, including the default administration shares if not required. Secure any remaining shares with restricted NTFS

permissions. Although shares may not be directly exposed to the Internet, a defense in depth strategy with limited and secured shares reduces risk if a server is compromised.

- **Ports**

  Unused ports are closed at the firewall, but servers behind the firewall also block or restrict ports based on their usage. For a dedicated SQL Server, block all ports except for the necessary SQL Server port and the ports required for authentication.

- **Registry**

  SQL Server maintains a number of security-related settings, including the configured authentication mode in the registry. Restrict and control access to the registry to prevent the unauthorized update of configuration settings, for example, to loosen security on the database server.

- **Auditing and Logging**

  Configure a minimum level of auditing for the database server using a combination of Windows and SQL Server auditing features.

- **SQL Server Security**

  A number of SQL Server security settings can be controlled through Enterprise Manager. These include the authentication mode, auditing level, and the accounts used to run the SQL Server service.

- **SQL Server Logins, Users, and Roles**

  SQL Server 2000 manages access control using logins, databases, users, and roles. Users (and applications) are granted access to SQL Server by way of a SQL server login. The login is associated with a database user and the database user is placed in one or more roles. The permissions granted to the role determine the tables the login can access and the types of operations the login can perform. This approach is used to create least privileged database accounts that have the minimum set of permissions necessary to allow them to perform their legitimate functionality.

- **SQL Server Database Objects**

  The ability to access SQL Server database objects, such as built-in stored procedures, extended stored procedures should be reviewed.

3.6.2 SQL Server Installation Considerations

Before taking steps to secure your database server, know the additional components that are present on a Windows 2000 or Windows Server 2003 server after SQL Server is installed.

What Does SQL Server Install?

When you install SQL Server, a number of Windows services are installed in addition to program and data files. By default, program and data files are located in the \Program Files\Microsoft SQL Server\ directory. The Table.1 shows the services and folders that are created.

Table 1: SQL Server Installation Defaults

Item	Details
Services	MSSQLSERVER MSSQLServerADHelper Microsoft Search SQLSERVERAGENT
Folders	\program files\Microsoft SQL Server\mssql\binn (program files) \program files\Microsoft SQL Server\mssql\data (data files including .mdf, .log, and .ndf) \program files\Microsoft SQL Server\80\tools (shared tools/books online) \program files\Microsoft SQL Server\mssql\logs (error logs) \program files\Microsoft SQL Server\mssql\backup (backup files) \program files\Microsoft SQL Server\mssql\jobs (temp job output files) For named instances, the instance name is used in the file path: \program files\Microsoft SQL Server\MSSQL$InstanceName\binn \program files\Microsoft SQL Server\MSSQL$InstanceName\data

Recommendations

It is a good idea to perform a custom installation of SQL Server so you can select the most secure installation options.

Installing SQL Server

When installing SQL Server on a production server, choose the custom setup option. You can selectively choose the items to install. You should not install the items listed in Table.2 on a production database server.

Table 2: Items Not to Install During Custom Installation

Tool	Purpose
Upgrade tools	Used to upgrade SQL Server 6.5 databases
Replication support	Script and binary files used for replication. (Do not install unless you need replication.)

Full text search	Full text search engine (Microsoft Search service). Do not install unless you require full text search.
Books online	SQL Server documentation
Development tools	Headers and library files used by C developers and Microsoft Data Access (MDAC), and XML software development kits (SDKs), and an interface for stored procedure debugging.
Code samples	Sample code used to educate developers.

Windows authentication offers the following advantages:

- Existing domain and local security policies can be used to enforce strong passwords and account management best practices.
- Credentials are not passed over the network.
- Application database connection strings do not require credentials.

If you select Mixed Mode, create a strong password for the sa account. The sa account is a prime target for password guessing and dictionary attacks.

3.6.3 Steps

The steps cover Windows 2000 and Windows Server 2003 and SQL Server 2000. Each step may contain one or more actions to secure a particular area or feature.

Step 1: Patches and Updates
Step 2: Services
Step 3: Protocols
Step 4: Accounts
Step 5: Files and Directories
Step 6: Shares
Step 7: Ports
Step 8: Registry
Step 9: Auditing and Logging
Step 10: SQL Server Security
Step 11: SQL Server Logins, Users and Roles
Step 12: SQL Server Database Objects

Step 1: Patches and Updates

Failure to apply the latest patches and updates in a timely manner means that you are providing opportunities for attackers to exploit known vulnerabilities. You should verify that your database server is updated with the latest Windows 2000 / Windows Server 2003 and SQL Server service packs and updates.

Step 2: Services

To reduce the attack surface area and to make sure you are not affected by undiscovered service vulnerabilities, disable any service that is not required. Run those services that remain using least privileged accounts.

Step 3: Protocols

Prevent the use of unnecessary protocols. Configure SQL Server to support only clients that connect using the TCP/IP protocol. Disable all other protocols, unless required.

Step 4: Accounts

Follow the principle of least privilege for the accounts and connect to SQL Server to restrict the capabilities of an attacker who manages to execute SQL commands on the database server. Also apply strong password.

Step 5: Files and Directories

Securing operating systems files using ACLs as well as harden NTFS permissions to restrict access to SQL Server program files, data files, and log files together with system level tools. Additionally, the SQL Server service account should have access only to what it needs.

Additional Considerations

To improve your database server security further:

- Remove unused applications that may be installed on the server.
- Encrypt your data files using Encrypting File System (EFS).

To implement EFS, right-click the directory, click **Advanced**, and then click **Encrypt contents to be secure**.

Step 6: Shares

Remove any unused shares and harden the NTFS permissions on any required shares. By default, all users have full control on newly created file shares. Harden these default permissions to make sure that only authorized users can access files exposed by the share.

Step 7: Ports

By default, SQL Server listens on TCP port 1433 and uses UDP port 1434 for client-server negotiation. Use a combination of firewalls and IPSec policies to restrict access to these ports to minimize the avenues of attack open to an attacker.

Step 8: Registry

When you install SQL Server, it creates a number of registry entries and subentries that maintain vital system configuration settings. It is important to secure these settings to prevent an attacker from changing them to compromise the security of SQL Server installation.

Step 9: Auditing and Logging

Auditing does not prevent system attacks, although it is a vital aid in identifying intruders, attacks in progress, and to diagnose attack footprints. It is important to enable all auditing mechanisms at your disposal, including Windows operating system level auditing and SQL Server login auditing.

Step 10: SQL Server Security

The settings discussed are configured using the **Security** tab of the **SQL Server Properties** dialog box in Enterprise Manager. The settings apply to all the databases in a single instance of SQL Server. The **SQL Server Properties** dialog box is shown in Fig. 1.

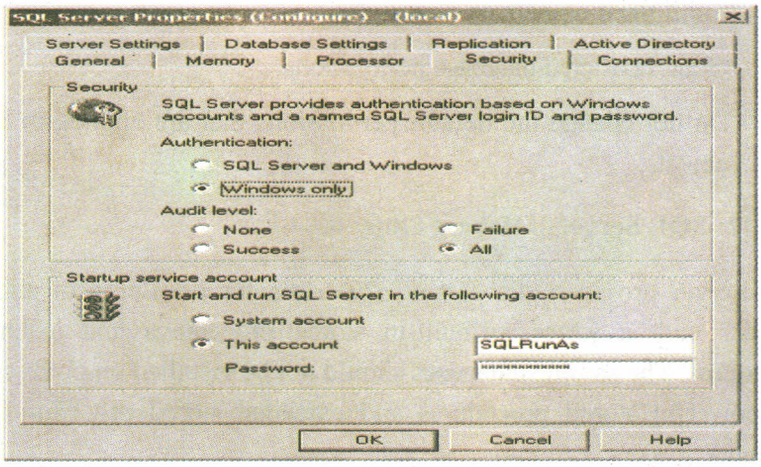


Fig.1: SQL Server security properties

Step 11: SQL Server Logins, Users and Roles

To be able to access objects in a database you need to pass two layers of security checks. First, you need to present a valid set of login credentials to SQL Server. If you use Windows authentication, you need to connect using a Windows account that has been granted a SQL Server login. If you use SQL Server authentication, you need to supply a valid user name and password combination.

The login grants you access to SQL Server. To access a database, the login must be associated with a database user inside the database you want to connect to. If the login is associated with a database user, the capabilities of the login inside the database are determined by the permissions associated with

that user. If a login is not associated with a specific database user, the capabilities of the login are determined by the permissions granted to the public role in the database. All valid logins are associated with the public role, which is present in every database and cannot be deleted. By default, the public role within any database that you create is not granted any permissions.

Use the following recommendations to improve authorization settings in the database:

- Use a strong sa (system administrator) password.
- Remove the SQL guest user account.
- Remove the BUILTIN\Administrators server login.
- Do not grant permissions for the public role.

Additional Considerations

Also consider the following recommendations when configuring SQL Server logins, users, and roles:

- Limit the members of sysadmin.
- Grant restricted database permissions.
- Do not change the default permissions that are applied to SQL Server objects.

Step 12: SQL Server Database Objects

SQL Server provides two sample databases for development and education together with a series of built-in stored procedures and extended stored procedures. The sample databases should not be installed on production servers and powerful stored procedures and extended stored procedures should be secured.

3.6.4 Staying Secure

Regularly monitor the security state of your database server and update it regularly to help prevent newly discovered vulnerabilities from being exploited. To help keep your database server secure:

- Perform regular backups.
- Audit group membership.
- Monitor audit logs.
- Stay current with service packs and patches.
- Perform security assessments.
- Use security notification services.

3.7 CODE REVIEW

Code reviews should be a regular part of development process. Security code reviews focus on identifying insecure coding techniques and vulnerabilities that could lead to security issues. The review goal is to identify as many potential security vulnerabilities as possible before the code is deployed. The cost and effort of fixing security flaws at development time is far less than fixing them later in the product deployment cycle.

Managed Code

Use the review questions in this section to analyze your entire managed source code base. The review questions apply regardless of the type of assembly. This section helps you identify common managed code vulnerabilities.

The following review questions help you to identify managed code vulnerabilities:

- Is your class design secure?
- Do you create threads?
- Do you use serialization?
- Do you use reflection?
- Do you handle exceptions?
- Do you use cryptography?
- Do you store secrets?
- Do you use delegates?

Code Access Security

All managed code is subject to code access security permission demands. Many of the issues are only apparent when your code is used in a partial trust environment, when either your code or the calling code is not granted full trust by code access security policy.

Use the following review points to check that you are using code access security appropriately and safely:

- Do you support partial-trust callers?
- Do you restrict access to public types and members?
- Do you use declarative security?
- Do you call Assert?
- Do you use permission demands when you should?

- Do you use link demands?
- Do you use Deny or PermitOnly?
- Do you use particularly dangerous permissions?
- Do you compile with the /unsafe option?

Data Access Code

This section identifies the key review points that should be considered while reviewing your data access code.

- Do you prevent SQL injection?
- Do you use Windows authentication?
- Do you secure database connection strings?
- How do you restrict unauthorized code?
- How do you secure sensitive data in the database?
- Do you handle ADO .NET exceptions?

Check Your Progress 1

Note: a) Space is given below for writing your answer.

b) Compare your answer with the one given at the end of the Unit.

1) Explain the steps required to secure a web server.

.....

.....

.....

.....

2) How to secure communication channel?

.....

.....

.....

.....

3) Explain the various database server configuration categories.

.....

.....

.....

.....

4) What is security code review and why is it important?

.....

.....

.....

.....

3.8 LET US SUM UP

The methodology used in this unit allows you to build a secure Web server from scratch and also allows you to harden the security configuration of an existing Web server. The next step is to ensure that any deployed applications are correctly configured. The unit focuses on the application server configuration and the associated communication channels that connect the Web server to the application server and the application server to the database server. Technologies covered include Enterprise Services, Web services etc. Database servers are a prime target for attackers. The database server must be secured against internal, external, network level, and application level attacks. A secure database server includes a hardened SQL Server 2000 installation on top of a hardened Windows 2000 / Windows Server 2003 installation, coupled with secure network defenses provided by routers and firewalls. Security code reviews are similar to regular code reviews or inspections except that the focus is on the identification of coding flaws that can lead to security vulnerabilities. The added benefit is that the elimination of security flaws often makes your code more robust.

3.9 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) The process of securing your Web server use the configuration categories introduced earlier in this chapter. Each high-level step contains one or more actions to secure a particular area or feature.

Step 1 Patches and Updates
Step 2 IIS Lockdown
Step 3 Services
Step 4 Protocols
Step 5 Accounts
Step 6 Files and Directories
Step 7 Shares
Step 8 Ports
Step 9 Registry
Step 10 Auditing and Logging
Step 11 Sites and Virtual Directories
Step 12 Script Mappings
Step 13 ISAPI Filters
Step 14 IIS Metabase
Step 15 Server Certificates
Step 16 Machine.config
Step 17 Code Access Security

2) Sensitive application data and authentication credentials that are sent to and from the application server should be encrypted to provide privacy and integrity. This mitigates the risk associated with eavesdropping and tampering. If you consider this threat to be negligible in your environment — for example, because your application is located in a closed and physically secured network — then you do not need to encrypt the traffic. If network eavesdropping is a concern, then you can use SSL, which provides a secure communication channel at the application layer, or IPSec, which provides a transport-level solution. IPSec encrypts all IP traffic that flows between two servers, while SSL allows each application to choose whether or not to provide an encrypted communication channel.

3) **Database server security categories**

The configuration categories shown in Figure 18.2 are based on best practices obtained from field experience, customer validation, and the study of secure deployments. The rationale behind the categories is as follows:

- Patches and Updates
- Services
- Protocols
- Accounts
- Files and Directories
- Shares
- Ports
- Registry
- Auditing and Logging
- SQL Server Security
- SQL Server Logins, Users, and Roles
- SQL Server Database Objects

4) Security code reviews are similar to regular code reviews or inspections except that the focus is on the identification of coding flaws that can lead to security vulnerabilities. The added benefit is that the elimination of security flaws often makes your code more robust. Security code reviews are not a panacea. However, they can be very effective and should feature as a regular milestone in the development life cycle.

3.10 SUGGESTED READINGS

- http://csrc.nist.gov/publications/nistpubs/800-123/SP800-123.pdf
- http://msdn.microsoft.com/
- http://www.indiamart.com/magnosolutions/services.html
- http://www.phyzertechnologies.com/appsecurity.aspx
- http://www.sans.org/reading_room/whitepapers/application/framework-secure-application-design-development_842

UNIT 4 APPLICATION THREAT MODELING

Structure

4.0 INTRODUCTION

Threat modeling is an approach for analyzing the security of an application. It is a structured approach that enables you to identify, quantify, and address the security risks associated with an application. Threat modeling is not an approach to reviewing code, but it does complement the security code review process. In developing software a software development life cycle is there in which steps are starting from requirement phase but there is a huge change in the concept and the development starts from threat that means the inclusion of threat modeling in the SDLC can help to ensure that applications are being developed with security built-in from the very beginning. This, combined with the documentation produced as part of the threat modeling process, can give the reviewer a greater understanding of the system. This allows the reviewer to

see where the entry points to the application are and the associated threats with each entry point. The concept of threat modeling is not new but there has been a clear mindset change in recent years because necessity is the mother of invention. Modern threat modeling looks at a system that how to protect the software from a potential attacker's perspective. Microsoft has made threat modeling a core component of their SDLC, which they claim to be one of the reasons for the increased security of their products in recent years.

When source code is analyzed outside the software development life cycle, threat modeling reduces the complexity of the software by promoting an in-depth first approach vs. breadth first approach. Instead of reviewing all source code with equal focus, a developer has to rank the threats according to the priority of risk.

It is also called an engineering technique which is used to help us in identifying threats, attacks, vulnerabilities and countermeasures in the context of application scenario. It also shape our application design to meet our security objectives and make trade-off during key engineering decisions and reduce risk of security issues arising during development and operations.

4.1 OBJECTIVES

After studying this unit, you should be able to:

- describe Entity Authentication Protocols;
- explain Key Establishment; and
- explain time stamping.

4.2 THE STEPS IN THREAT MODELING PROCESS

The policy is divide and rule that means developer has to divide the application in parts according to risks that is why the threat modeling process can be decomposed into 3 high level steps:

Step 1:

(1) Decompose the Application.

(2) Understand the application well.

(3) Understanding how it reacts with external entities.

(4) Identify the entry points and assets where attacker can attack.

(5) Identify the trust level to whom we have to give access rights of code.

(6) Check all the threats to the system using Dataflow Diagrams so that the flow of the program should be clear.

Step 2:

(1) Find and rank all the threats according to how the attacker can attack and how the defender can protect.

(2) A threat categorization such as STRIDE (spoofing, tampering, Repudiation, Information Disclosure, Denial of service, Elevation of privilege) can be used to rank the threats.

(3) Threat categories such as Auditing and logging, authentication i.e Application security Frame can be used to rank the threats.

(4) Data Flow Diagram (DFD) helps us to find the attacker view i.e. from where the attacker is going to attack and how to prevent the software.

(5) Threats are also called as weakness of the security controls.

(6) Use case diagrams also help us to find where the system is weak and where the attacker can attack.

(7) This determination of security risk for each threat can be determined using a value based risk model DREAD(Damage, Reproducibility, Exploitability, Affected users, discoverability)

Step 3:

(1) Determine countermeasures and mitigation.

(2) A lack of protection against a threat causes vulnerability whose risk exposure could be mitigated with the implementation of a countermeasure.

(3) When the threats are ranked sort the threats from higher rank to lower so that protective Measure can be taken.

(4) Prioritize the mitigation effort.

(5) Inform the user about the threat so that threat can be prevented.

Each of the above steps is documented as they are carried out. The resulting document is the threat model for the application. This guide will use an example to help explain the concepts behind threat modeling. The example that will be used is a college library website. The threat model for the college library website is produced. The steps of college library website threats are given. For example when anything is developed threats are involved in the same way when the package of library website is developed threats are involved. Student's staff and library staff are the main user of the library

website. Students have to get the books from library they will be issued user-id and passwords for using but user-id and log ins will be copied or hacked to get the books from library so these types of threats are available. A developer has to give access rights to every user according to their needs. The rights are different for different category. In the beginning people does not think about the security of the systems but a need arise when threats are developed. A developer has to make the system secure so that the user feels secure to use the system.

For Example in the use of credit card or debit card system if some one has to pay the payment through credit card or debit card threats are involved. If some one can copy the credit card no and know the password and he can operate with that card but security measures are used everywhere. When ever the card is used a message is generated in the mobile as a security measure to overcome the threat. If security is not there then nobody is going to use the plastic money. In every step of using plastic money in any type security step is involved in every step to overcome the threat.

4.3 ENTITY AUTHENTICATION PROTOCOLS

Security Issues and Authentication

In computer Security, authentication is the process of attempting to verify the digital identity of an entity. This is done with the help of a username and a password. Security is one place where the world of computation interacts with the physical world. Nothing proves this better than the fact that most security breaches in connection with computers are based on "social engineering" rather than flaws in encryption, access control or authentication algorithms. Hence, our first step in understanding authentication must be to understand the context in which authentication takes place. We will then examine the different mechanisms by which authentication is achieved and finally we will see how authentication can be translated into access and authorization.

4.3.1 The Context

Authentication is the process of determining whether an identity is permitted to perform some action, such as accessing a resource. In order to authenticate we must first grant anonymous access to a certain resource which has an immutable attribute. Authentication is the process of identifying the entity that has obtained access to this resource.

In the "distant" past when the only way in which multi-tasking multi-user systems were used was via terminals that were permanently attached to the computer. These terminals were placed in individual offices of the users. At this point authentication was not required since one could assume that only person with access to a specific office was a specific user of the resources. The

user was identified by the resource he made use of. In this case the resource and its immutable attribute is the terminal which is an input/output device for the computer. The entity is identified by the physical access that the user has made to that resource.

So what happens when user B walks into user A's office and wants to use the computer from there? The solution is to introduce the username: prompt. The user can get the authentication there by getting authorization. Each terminal has a reset button. When this button is pressed the computer prompts the user to supply a name and then authenticates the user by the name supplied.

In the previous case the identification was based on the trust of the individual whose office is used by the second user; the system did not even ask for a password. But, when multi-user systems had terminals in public access areas passwords were required; at this point access to the terminal had truly become anonymous and thus some form of computer-based authentication was required to make the system secure.

In this case, when the computer had no control over the data that was flowing in via this terminal even before authentication, had been performed. Thus the program login that monitored the logins had to handle this incoming data carefully otherwise there could be buffer overflow attacks, denial of service attacks and so on. Handling "tainted" data from an unauthenticated source is an important facet of all programs that deal with anonymous entities; e. g. authentication programs.

Once network connectivity is established, the possibility of the terminal not being attached to the computer directly but over an external wire. The immutable attribute that was algorithmically authenticated was now limited to a serial port. In this case, one must assume that the physical equipment i.e the wire was inviolable.

However, in practice it could be tapped or spliced. This became more serious when packet-switched networks were used in place of the external wire. The terminal oriented connection protocol or TCP was designed to provide an immutable attribute called a TCP socket. In this case establishing a TCP connection involved the creation of a software "socket" which is represented as a pair of pairs. Here the pair consists of a host ID and a port. The socket is authenticated by the authentication process.

However, this creation of a socket out of a packet-switched network protocol (IP) is quite problem-ridden and requires its own host-authentication methods. In the past, simple host-authentication methods using IP addresses were considered sufficient. Nowadays there is a move towards securing the networks using SSL, IPSec or more general authentication techniques present in IPv6. In this case, there is low-level authentication via session-keys and other mechanisms that permeates the entire transaction in a manner that is mostly

invisible to the user. Such techniques make use of immutable attributes that are mostly software; but one may ask "Can one really make something immutable out of software?" That we can is the legend of cryptographic authentication.

4.3.2 Mechanisms

As explained above the first authentication mechanism used is the just a "username" or IP address. Since the external entity provides this information at the time that it acquires anonymous access, it may or may not be reliable for identification depending on the physical context. So it is common to have an additional proof. One of the standard mechanisms used is a password.

In the older contexts where the possibility of wire tapping or wire splicing was remote, such a password would be transmitted "in the clear" over the network. Nowadays, that is not considered particularly secure unless the channel is secured in some way. However, securing the channel requires some other authentication in order to prevent monkey-in-the-middle attacks. In either case we need to consider some form of authentication that does not involve the clear transmission of the password.

In spite of this, a number of systems around the world still use password-based authentication or something similar—like requiring the user to present a credit-card number or date of birth. Other than wire-tapping and replay attacks, this has other vulnerabilities. For example, users often have the same password on all systems; if the "password" is biometric then this may not even be the user's choice! The information is "given away" by the user to the authenticating system—but not all systems that wish to authenticate you are reliable; e. g. we have seen a spate of "phishing" attacks. So what are the replacements for password based systems?

One way is to use a different password each time. In a "one-time-password" system a password can only be used once. The obvious difficulty with this system is how the user and the system decide on the sequence or list of passwords to be used. Another difficulty is that if the user does not verify the system that requests the password. So this is still susceptible to "phishing" and monkey-in-the-middle even though the damage may be limited to one session. One significant advantage of this system is that the user is not required to do any computations unlike the systems being discussed below.

Another way to generate the list of one-time-passwords is to use a secret generating system. This second secret must be shared between the user and the service provider so we have "shared secret" systems. In this case the authenticating system and the user have a shared secret. This secret can also be used to create some kind of time-stamped token which is exchanged as proof of possession of the shared secret; the actual secret is never exchanged across the wire. For example, we could take a cryptographic hash function and apply it to the concatenation of the current time, the local address and the shared secret;

each party sends its own version of this across the wire. In such a system both the user and the system are sure of the identity of the other party (assuming the invulnerability of the cryptosystem used). Vulnerability is at the point where the shared secret is created and stored by the user and the system. Another vulnerability is due to the fact that today's user must interact with a number of different systems and so must keep a shared secret with each of these systems; the larger the amount of secret material to be stored the greater its vulnerability.

An alternative is the public-key system. Each user and each system that participate in the protocol publish the public-key component of a private-public pair. The cryptographic invulnerability of the cryptosystem will mean that only the entity which generated the public key can have the private key—unless that entity is compromised. Each party uses the private key to attach a "digital signature" to a concatenation of the current time and the local socket address; the other party can then verify the signature and thus authenticate the other party. A major vulnerability of this system is the difficulty of verifying that the public-key actually belongs to the entity whom one wants to authorize. Usually public-keys are exchanged at a "key-signing party" or via a tree or web of trust in order to circumvent this problem. Another vulnerability of this system is the requirement of significant computational resources for the creation and verification of digital signatures; this can be used to initiate a denial of service attack.

One way around this computational requirement is the "trusted third party" authentication system. In this case, there is a separate entity that everyone trusts and has significant computational resources dedicated to providing others with authentication information. An elaborate protocol called "Kerberos" has been developed that explains how a session may be securely initiated between parties who are individually known to this Authenticator. Kerberos is designed to address the problem of authentication in a network of slightly trusted client systems. Each party expends some computational resources in obtaining recognition from the Authenticator but does not need to spend significant computational resources thereafter.

4.3.3 Applications

Once an input resource like a socket is authenticated then all data that comes in from that socket can be marked as coming from that particular authenticated source and valid data. The login application that we started makes essentially use of this authentication. However, off late numerous other applications also make use of authentication; e. g. web servers. In all such cases it is important to distinguish between authentication and authorization. An authorization means a person is authorized to work on the system.

The process that carries out the authentication transaction does not need to have access to too many system resources but in the case of password-based system it needs to have access only to the appropriate form of the password database. Sometime the authenticator can run the process with low privileges also. In particular, compromise of the authenticator does not automatically grant access at all levels and such a compromise can achieve is fake authentication.

On the other hand a process can only grant access to those resources that the processor can access easily. Thus the authorizing process must have very wide access to the system. For example, the login process must run with "root" privileges after it has authenticated the user and is creating a shell process for a particular user. In a typical scenario, an application runs a number of sub-processes at lower privilege; one of these is the authentication process and the rest are authenticated processes. The later processes run with the privilege level of the user who has been authorized by the authentication process. It is ideal if the top-level of the application limits its "job" to the creation of these processes as that simplifies its security analysis. Since this top-level is the authorizing process, its compromise would lead to a wide-range compromise of the system.

In practice one sees that monolithic programming is the norm and the kind of privilege separation that is recommended above is only slowly making its way into application programming. One can hope that this situation will improve as the different roles of authentication and authorization become clearer to programmers.

Security (cryptographic protocol or encryption protocol) is an abstract or concrete protocol that performs a security-related function and applies cryptographic methods. Cryptography is the science of using mathematics to encrypt and decrypt data. It is the art of secret writing. A protocol describes how algorithms should be used. A sufficiently detailed protocol includes details about data structures and representations, at which point it can be used to implement multiple, interoperable versions of programs. Cryptographic protocols are widely used for secure application-level data transport. **A cryptographic protocol** usually incorporates at least some of these aspects:

- Key agreement or establishment
- Entity Authentication
- Non-repudiation method

Entity authentication protocols verify the credibility of incoming messages from one network to another network. Techniques like cryptography and encryption are used to develop these protocols. The main aim behind the protocol development is to avoid hacking issues in many business industries such as finance and e-commerce.

4.3.4 LCMQ Entity Authentication Protocol

LCMQ Entity Authentication Protocol is a lightweight unconventional, secure entity authentication scheme. It is used for radio frequency identification (RFID) systems. RFID systems consist of simple, low cost tags that are attached to physical objects and powerful readers that queue data from tags. Sometimes radio frequency transmissions are used for identification of physical entities that is why these RFID systems are used in many applications such as inventory monitoring, payment, electronic password. In big stores bar codes are used for items placed in the store and codes are read with the radio frequency waves for taking the rates calculation and the bills are generated.

4.3.5 ECC Entity Authentication Protocol

ECC Entity Authentication Protocol are designed to secure the information and keeps the secret for communication. The rapid growth of computer applications for exchanging information electronically has resulted in the elimination of physical ways for providing security through locks, sealing and signing documents. This has thus resulted in the need for techniques for securing electronic documents transactions with the help of ECC Entity Authentication Protocol. The science of keeping messages secure is called cryptography. ECC Entity Authentication protocols are used to get operations in credit cards, debit cards, and mobile phones etc. ECC analyzes the error codes and verifies the codes from its database to operate accordingly. The protocol can resend, reject, terminate and accept packets from other source.

4.3.6 3PAKE Entity Authentication Protocol

It is known as three party password based Authenticated key exchange Protocol (3 party PAKE) or three-party Password Exchange Protocol It is attractive due to their convenience in many communication applications with security. When two entities are communicating on the internet the convenience of password based authentication is must. It is the method of doing secure communication on either side. Now with the development of wireless communication for the internet the value of this secure protocol increased. When the server is acting as 3 PAKE server holding all the registered users passwords and plays a role of authentication agent.

Check Your Progress 1

1) What is non repudiation?

.....

.....

.....

.....

2) Define the term Authentication.

.....

.....

.....

.....

3) What is LCMQ Entity Authentication Protocol?

.....

.....

.....

.....

4.4 KEY ESTABLISHMENT

Key establishment is a technique or a protocol using which a secret can be available to two or more parties, for subsequent cryptographic use. Key establishment may be broadly subdivided into two parts:-

- key transport
- key agreement

A **key transport protocol** or mechanisms is a key establishment technique where one party generates or obtains a secret value by some means, and securely transfers it to the other(s).

A **key agreement protocol** or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value. Key establishment protocols involving authentication typically require a set-up phase whereby authentic and possibly secret initial keying material is distributed. Most protocols have as an objective the creation of distinct keys on each protocol execution. In some cases, the initial keying material pre-defines a fixed key which will result every time the protocol is executed by a given pair or group of users. Systems involving such static keys are insecure under known-key attacks.

Besides this some Key pre-distribution schemes are key establishment protocols whereby the resulting established keys are completely determined a prior by initial keying material. In contrast, dynamic key establishment schemes are those whereby the key established by a fixed pair (or group) of users varies on subsequent executions.

Dynamic key establishment is also referred to as session key establishment. In this case the session keys are dynamic, and it is usually intended that the protocols are immune to known-key attacks.

Many key establishment protocols involve a centralized or trusted party, for either or both initial system setup and on-line actions (i.e., involving real-time participation). This party is referred to by a variety of names depending on the role played, including: trusted third party, trusted server, authentication server, key distribution center (KDC), key translation center (KTC), and certification authority (CA).

4.4.1 Entity Authentication, Key Authentication

It is generally desired that each party in a key establishment protocol be able to determine the true identity of the other(s) which could possibly gain access to the resulting key, implying avoidance of any unauthorized additional parties from deducing the same key. In this case, the technique is said (informally) to provide secure key establishment. This requires both secrecy of the key, and identification of those parties with access to it. Furthermore, the identification requirement differs subtly, but in a very important manner, from that of entity authentication – here the requirement is knowledge of the identity of parties which may gain access to the key, rather than justification that actual communication has been established with such parties.

Key authentication is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key. Key authentication is independent of the actual possession of such key by the second party, or knowledge of such actual possession by the first party; in fact, it need not involve any action whatsoever by the second party. For this reason, it is sometimes referred to more precisely as (implicit) key authentication. Key confirmation is the property whereby one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key. Explicit key authentication is the property obtained when both (implicit) key authentication and key confirmation hold.

In the case of explicit key authentication, an identified party is known to actually possess a specified key, a conclusion which cannot otherwise be drawn. Encryption applications utilizing key establishment protocols which offer only implicit key authentication often begin encryption with an initial known data unit serving as an integrity check-word, thus moving the burden of key confirmation from the establishment mechanism to the application. The focus in key authentication is the identity of the second party rather than the value of the key, whereas in key confirmation the opposite is true. Key confirmation typically involves one party receiving a message from second containing evidence demonstrating the latter's possession of the key. In

practice, possession of a key may be demonstrated by various means, including producing a one-way hash of the key itself, use of the key in a (keyed) hash function, and encryption of a known quantity using the key. These techniques may reveal some information (albeit possibly of no practical consequence) about the value of the key itself; in contrast, methods using zero-knowledge techniques allow demonstration of possession of a key while providing no additional information (beyond that previously known) regarding its value.

Entity authentication is not a requirement in all protocols. Some key establishment protocols provide none of entity authentication, key authentication, and key confirmation. Unilateral key confirmation may always be added e.g., by including a one-way hash of the derived key in a final message.

In a key establishment protocol which involves entity authentication, it is critical that the protocol be constructed to guarantee that the party whose identity is thereby corroborated is the same party with which the key is established. When this is not so, an adversary may enlist the aid of an unsuspecting authorized party to carry out the authentication aspect, and then impersonate that party in key establishment

4.4.2 Assumptions in Key Establishment Protocols

To clarify the threats protocols may be subject to, and to motivate the need for specific protocol characteristics, one requires (as a minimum) an informal model for key establishment protocols, including an understanding of underlying assumptions. Attention here is restricted to two-party protocols, although the definitions and models may be generalized.

4.4.3 Adversaries in Key Establishment Protocols

Communicating parties or entities in key establishment protocols are formally called principles, and assumed to have unique names. In addition to legitimate parties, the presence of an unauthorized "third" party is hypothesized, which is given many names under various Circumstances, including: adversary, intruder, opponent, enemy, attacker, eavesdropper, and impersonator. When examining the security of protocols, it is assumed that the underlying cryptographic mechanisms are used, such as encryption algorithms and digital signatures schemes, are secure. If otherwise, then there is no hope of a secure protocol. An adversary is hypothesized to be not a cryptanalyst attacking the underlying mechanisms directly, but rather one attempting to challenge the protocol objectives by defeating the manner in which such mechanisms are combined, i.e., attacking the protocol itself.

4.4.4 Secret Sharing

Secret sharing is a method for distributing a secret amongst a group of participants, each of which is allocated a share of the secret. The secret can only be reconstructed when the shares are combined together and individual shares are of no use. Secret sharing schemes are multi-party protocols related to key establishment. The original motivation for secret sharing is given. To safeguard cryptographic keys from loss, it is desirable to create backup copies. The greater the number of copies made, the greater the risk of security exposure; the smaller the number, the greater the risk that all are lost. Secret sharing schemes address this issue by allowing enhanced reliability without increased risk. They also facilitate distributed trust or shared control for critical activities (e.g., signing corporate cheques; opening bank vaults), by gating the critical action on cooperation by t of n users. The idea of secret sharing is to start with a secret, and divide it into pieces called shares which are distributed amongst users such that the pooled shares of specific subsets of users allow reconstruction of the original secret. This may be viewed as a key pre-distribution technique, facilitating one-time key establishment, wherein the recovered key is pre-determined (static), and, in the basic case, the same for all groups. A secret sharing scheme may serve as a shared control scheme if inputs (shares) from two or more users are required to enable a critical action

4.4.5 Conference Keying

A **conference keying protocol** is a generalization of two-party key establishment to provide three or more parties with a shared secret key. Despite superficial resemblance, conference keying protocols differ from dynamic secret sharing schemes in fundamental aspects. General requirements for conference keying include that distinct groups recover distinct keys (session keys); that session keys are dynamic (excepting key pre-distribution schemes); that the information exchanged between parties is non-secret and transferred over open channels; and that each party individually computes the session key (vs. pooling shares in a black box). A typical application is telephone conference calls. The group able to compute a session key is called the privileged subset. When a central point enables members of a (typically large) privileged subset to share a key by broadcasting one or more messages, the process resembles pre-positioned secret sharing somewhat and is called broadcast encryption. An obvious method to establish a conference key K for a set of t _ 3 parties is to arrange that each party share a unique symmetric key with a common trusted party. Thereafter the trusted party may choose a new random key and distribute it by symmetric key transport individually to each member of the conference group. Disadvantages of this approach include the requirement of an on-line trusted third party, and the communication and computational burden on this party. A related approach not requiring a trusted party involves a designated group member (the chair) choosing a key K,

computing pairwise Diffie-Hellman keys with each other group member, and using such keys to securely send K individually to each. A drawback of this approach is the communication and computational burden on the chair, and the lack of protocol symmetry.

4.5 TIME STAMP

Time stamping is recording the time of creation or existence of information. A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred. It represents the time at which an event is recorded by a computer, not the time of the event itself. **Time stamping** is the process of safely keeping track of the creation and modification time of a document. Basically time stamping is needed to properly validate the signature. Time stamping should be used if the signature is supposed to be used in long term, i.e. longer than one or several days. But Time stamping is not necessary when we, for example, send a short signed note to the colleague and this note is expected to be read and disposed of the same day as it has been written. Of course, time stamping can not be used when it's not supported by the signing technologies or when time stamping authority is not available.

4.5.1 How Time Stamping Works?

Time stamping involves two components. The code to be time stamped is treated as time stamping client and a time stamping server called Time Stamping Authority (TSA).Firstly the code signs some data and then the hash of the data signature is calculated. This hash is sent to TSA for signing. TSA signs the received hash using TSA certificate and includes current time on the server to this signature. The signature made by TSA is sent back to the code and the code adds this signature to the original signature made over the initial data. TSA services are offered by the companies which issue digital certificates Besides this there exist local (national, governmental or private) time stamping authorities, but their usability is limited as they are usually offered as part of some closed infrastructure. For example, if the governmental agency or bank accepts digitally signed documents, it can offer the TSA for use with such documents, but this TSA will be only accepted and validated by this governmental agency or bank.

It must be notice that as like any other signature, time stamping signature can become invalid if the time stamping certificate expired.

4.5.2 Importance of Time Stamp

The timestamp tells the entity, when exactly the signature was made and validates the signature. As we know that the certificate is not everlasting. It has certain validity period, i.e. the certificate may only be used for it's purpose during some period of time. If we use the certificate, that has expired, to sign

the data, such signature will not be accepted as valid. If the signature validator finds a timestamp, it will know when the signature was made, and will check if the certificate was valid at that moment of time. If there's no timestamp, then nobody knows, when the signature was made, and it's assumed that it could be made at any moment of time, possibly after the certificate has expired. There are two possible results from this situation: either the signature is claimed as not valid, or the signature is assumed to be made at the moment of validation. In the second case, if the signing certificate itself has expired by the moment of signature validation, the signature will not be accepted as valid too. And if the signature is expected to be validated somewhere in future, then it's likely that such problem will happen sooner or later. So if the signature is not time stamped properly, there are chances that it will not be accepted as valid.

4.5.3 Classification of Time Stamping

The idea of time stamping information was very old and it can be classified into a number of categories depending upon the objectives of security purpose to overcome the threats and make the system secure.

- PKI-based - Timestamp
- Linking-based schemes
- Transient key scheme
- MAC - simple secret key based scheme

PKI Based Time Stamp

PKI digital signature protects time stamp token. Firstly a hash is calculated from the given data. A hash is a sort of digital fingerprint of the original data: a string of bits that is different for each set of data. If the original data is changed then this will result in a completely different hash. This hash is sent to the Time stamping Authority (TSA). The TSA concatenates a timestamp to the hash and calculates the hash of this concatenation. This hash is in turn digitally signed with the private key of the TSA. This signed hash + the timestamp is sent back to the requester of the timestamp who stores these with the original data Since the original data cannot be calculated from the hash, the TSA never gets to see the original data, which allows the use of this method for confidential data

Linking-Based Schemes

Linking-based time-stamping is a type of time stamping where issued time-stamps are related to each other.

This time-stamping creates time-stamp tokens which are dependent on each other, mixed into some authenticated data structure. Later modification of issued time-stamps would invalidate this structure. Temporal order of issued

time-stamps is also protected by this data structure, making backdating of the issued time-stamps is impossible, even by the issuing server itself. Top of the authenticated data structure is generally published in some hard-to-modify and widely witnessed media like printed newspaper

Suitable candidates for authenticated data structure are:

- Linear hash chain,
- Hash tree
- Skip list.

The linking-based time-stamping authority (TSA) usually performs the following distinct functions:

Aggregation

For increased scalability TSA might group time-stamping requests arriving within a short timeframe. These requests will be aggregated together without retaining their temporal order and then assigned the same time value. Aggregation creates cryptographic connection between all involved requests; authenticating aggregate value will be used as input for the linking operation.

Linking

Linking creates verifiable and ordered cryptographic link between current and already issued time-stamp tokens.

Publishing

TSA publishes periodically some links, so that all previously issued time-stamp tokens depend on the published link and that it is practically impossible to forge the published values. By publishing widely witnessed links the TSA creates unforgettable verification points for validating all previously issued time-stamps.

Security Aspects

Linking-based time-stamping is more secure than the usual, public-key signature based time-stamping. All consequential time-stamps "seal" previously issued ones - hash chain, could be built only in one way; modifying issued time-stamps is nearly as hard as finding a pre image for the used cryptographic hash function.

Linking-based time-stamping scales well - hashing is much faster than public key cryptography. There is no need for specific cryptographic hardware with its limitations.

Transient Key Scheme

In this system sessions are generated for short period and then destroyed with in the time limit to make the system secure and In a transient-key system, private keys are used for a short period and then destroyed, that is why it is sometimes called "**disposable crypto**." Data encrypted with a private key associated with a specific time interval can be linked to that interval, making transient-key cryptography particularly useful for digital time stamping. In a transient-key system, the source of time must be a consistent standard understood by all senders and receivers. Since a local system clock may be changed by a user, it is never used as a source of time. Instead, data is digitally signed with a time value derived from Universal Coordinated Time (UTC) accurate to within a millisecond. Whenever a time interval in a transient-key system expires, a new public/private key pair is generated, and the private key from the previous interval is used to digitally certify the new public key. The old private key is then destroyed.

MAC - Simple Secret Key Based Scheme

A **message authentication code** (**MAC**) is a short piece of information used to authenticate a message. A MAC algorithm, sometimes called a **keyed** hash **function**, accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC (sometimes known as a tag). The MAC value protects both a message's data integrity as well as its authenticity, by allowing verifiers to detect any changes to the message content.

Security Aspects

MAC is computed over compressed data using a shared secret key and is responsible for the verification of integrity of the message included in the transmitted record and MAC functions are similar to cryptographic hash functions, but still they possess different security requirements. To be considered secure, a MAC function must resist existential forgery. This means that even if an attacker has an access to a document which possesses the secret key and generates MACs for messages, but the attacker cannot guess the MAC for other messages without performing infeasible amounts of computation.

MACs differ from digital signatures as MAC values are both generated and verified using the same secret key. This implies that the sender and receiver of a message must agree on the same key before initiating communications, as is the case with symmetric encryption. For the same reason, MACs do not provide the property of non-repudiation offered by signatures specifically in the case of a network-wide shared secret key: any user who can verify a MAC is also capable of generating MACs for other messages. In contrast, a digital signature is generated using the private key of a key pair, which is asymmetric encryption. Since this private key is only accessible to its holder, a digital

signature proves that a document was signed by none other than that holder. Thus, digital signatures do offer non-repudiation.

Check Your Progress 2

1) Define Key Establishment.

.....

.....

.....

.....

2) Define ECC Entity Authentication Protocol?

.....

.....

.....

.....

3) What do you mean by key agreement protocol?

.....

.....

.....

.....

4) What do you mean by Time Stamp?

.....

.....

.....

.....

5) Discuss the working of a Time Stamping.

.....

.....

.....

.....

6) What is PKI based time stamp?

.....

.....

.....

.....

7) Discuss linking based time stamping.

.....

.....

.....

.....

8) Discuss various security aspects of MAC.

.....

.....

.....

.....

4.6 LET US SUM UP

The most important issues to be considered in security are privacy, authentication, integrity and non repudiation. Privacy can be obtained by encryption of the plain text into cipher text. Authentication, integrity and non repudiation are achieved by using the digital signature. It covers entity authentication protocol, key establishment and time stamping technique. The study of security standard protocols is necessary for the security of sensitive information so that the authorized user can not access the sensitive data. In addition to this the entity authentication protocols are responsible for identifying the correct entity to whom and by whom the information is sent or received. At the same time with the help of these protocols a sender could not deny later, that the information was not send by him which was actually send by him. Using the concept of time stamping an eye can be kept at the time of development of a document. So in order to maintain the security issues, different techniques can be used so that the document should be authorized received without any tempering.

4.7 CHECK YOUR PROGRESS: THE KEY

Check Your Progress 1

1) Non-repudiation means that a receiver can prove that the received message came from a specific sender. In no case a sender can deny sending a message that is actually send by him. More specifically non-repudiation of origin, is an important aspect of digital signatures. By this property an entity that has signed some information cannot at a later time deny having signed it. Similarly, access to the public key only does not enable a fraudulent party to fake a valid signature.

2) **Authentication** means that the receiver of the message is sure of the sender identity and no other person sends the information except the real sender. People can be authenticated by recognizing their faces, features, voices and handwriting, although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context. For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

3) **LCMQ Entity Authentication Protocol** is a lightweight authentication protocol that filters the network packages and inspects the address in the back end. In large networks, the volume of messages is quite high to authenticate them. LCMQ is an advanced protocol that works on Object Oriented Concept. The algorithm of the protocol is based on a strong encryption method. It operates in multiple threads to inspect millions of messages in a single instance. It was developed on HB entity protocol standard. The protocol is a better solution in terms of storage expenses, cost of communication and overhead of tag computation.

Check Your Progress 2

1) Key establishment is a technique or a protocol using which a secret can be available to two or more parties, for subsequent cryptographic use. Key establishment may be broadly subdivided into two parts

   - key transport
   - key agreement,

   A **key transport protocol** or mechanisms is a key establishment technique where one party generates or obtains a secret value by some means, and securely transfers it to the other(s).

   A **key agreement protocol** or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value..

2) **ECC Entity Authentication Protocol** is a customized protocol to detect the error codes of authentication and propose solution accordingly. The protocol has a strong encryption algorithm and contributes significantly in

authenticating the RFID packets and releasing the correct information across other networks. The protocol plays a significant role in boosting server security and reducing costs and network load with its lightweight structure. ECC analyzes the error codes and verifies the codes from its database to operate accordingly. The protocol can resend, reject, terminate and accept packets from other source.

3) A **key agreement protocol** or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these, (ideally) such that no party can predetermine the resulting value. Key establishment protocols involving authentication typically require a set-up phase whereby authentic and possibly secret initial keying material is distributed. Most protocols have as an objective the creation of distinct keys on each protocol execution. In some cases, the initial keying material pre-defines a fixed key which will result every time the protocol is executed by a given pair or group of users. Systems involving such static keys are insecure under known-key attacks.

4) A **Time Stamp** is a sequence of characters, denoting the date and/or time at which a certain event occurred. It represents the time at which an event is recorded by a computer, not the time of the event itself. **Time stamping** is the process of safely keeping track of the creation and modification time of a document. Basically time stamping is needed to properly validate the signature. Time stamping should be used if the signature is supposed to be used in long term, i.e. longer than one or several days. But Time stamping is not necessary when we, for example, send a short signed note to the colleague and this note is expected to be read and disposed of the same day as it has been written. Of course, time stamping can not be used when it's not supported by the signing technologies or when time stamping authority is not available.

5) Time stamping involves two components. The code to be time stamped is treated as time stamping client and a time stamping server called Time Stamping Authority (TSA).Firstly the code signs some data and then the hash of the data signature is calculated. This hash is sent to TSA for signing. TSA signs the received hash using TSA certificate and includes current time on the server to this signature. The signature made by TSA is sent back to the code and the code adds this signature to the original signature made over the initial data. TSA services are offered by the companies which issue digital certificates Besides this there exist local (national, governmental or private) time stamping authorities, but their usability is limited as they are usually offered as part of some closed infrastructure. For example, if the governmental agency or bank accepts digitally signed documents, it can offer the TSA for use with such

documents, but this TSA will be only accepted and validated by this governmental agency or bank.

It must be notice that as like any other signature, time stamping signature can become invalid if the time stamping certificate expired.

6) Here time stamp token is protected using a PKI digital signature. Firstly a hash is calculated from the given data. A hash is a sort of digital fingerprint of the original data: a string of bits that is different for each set of data. If the original data is changed then this will result in a completely different hash. This hash is sent to the Time stamping Authority (TSA). The TSA concatenates a timestamp to the hash and calculates the hash of this concatenation. This hash is in turn digitally signed with the private key of the TSA. This signed hash + the timestamp is sent back to the requester of the timestamp who stores these with the original data Since the original data can not be calculated from the hash, the TSA never gets to see the original data, which allows the use of this method for confidential data.

7) **Linking-based time-stamping** is a type of time stamping where issued time-stamps are related to each other.

   This time-stamping creates time-stamp tokens which are dependent on each other, mixed into some authenticated data structure. Later modification of issued time-stamps would invalidate this structure. Temporal order of issued time-stamps is also protected by this data structure, making backdating of the issued time-stamps is impossible, even by the issuing server itself. Top of the authenticated data structure is generally published in some hard-to-modify and widely witnessed media like printed newspaper.

8) Although MAC functions are similar to cryptographic hash functions, but still they possess different security requirements. To be considered secure, a MAC function must resist existential forgery. This means that even if an attacker has an access to a document which possesses the secret key and generates MACs for messages, but the attacker cannot guess the MAC for other messages without performing infeasible amounts of computation. MACs differ from digital signatures as MAC values are both generated and verified using the same secret key. This implies that the sender and receiver of a message must agree on the same key before initiating communications, as is the case with symmetric encryption. For the same reason, MACs do not provide the property of non-repudiation offered by signatures specifically in the case of a network-wide shared secret key: any user who can verify a MAC is also capable of generating MACs for other messages. In contrast, a digital signature is generated using the private key of a key pair, which is asymmetric encryption. Since this private key is only accessible to its holder, a digital signature proves that a document was

signed by none other than that holder. Thus, digital signatures do offer non-repudiation.

4.8 SUGGESTED READINGS

- Forouzan, Behrouz A. *Data Communication and Networking.*
- Katz, J. and Lindell, Y. (2007). *Introduction to Modern Cryptography,* Chapman & Hall/CRC Press.
- Secrets & Lies: Digital Security in a Networked World

Student Satisfaction Survey

Student Satisfaction Survey of IGNOU Students

ignou THE PEOPLE'S UNIVERSITY

Enrollment No.	
Mobile No.	
Name	
Programme of Study	
Year of Enrolment	
Age Group	☐ Below 30 ☐ 31-40 ☐ 41-50 ☐ 51 and above
Gender	☐ Male ☐ Female
Regional Centre	
States	
Study Center Code	

Please indicate how much you are satisfied or dissatisfied with the following statements

Sl. No.	Questions	Very Satisfied	Satisfied	Average	Dissati-sfied	Very Dissati-sfied
1.	Concepts are clearly explained in the printed learning material	☐	☐	☐	☐	☐
2.	The learning materials were received in time	☐	☐	☐	☐	☐
3.	Supplementary study materials (like video/audio) available	☐	☐	☐	☐	☐
4.	Academic counselors explain the concepts clearly	☐	☐	☐	☐	☐
5.	The counseling sessions were interactive	☐	☐	☐	☐	☐
6.	Changes in the counseling schedule were communicated to you on time	☐	☐	☐	☐	☐
7.	Examination procedures were clearly given to you	☐	☐	☐	☐	☐
8.	Personnel in the study centers are helpful	☐	☐	☐	☐	☐
9.	Academic counseling sessions are well organized	☐	☐	☐	☐	☐
10.	Studying the programme/course provide the knowledge of the subject	☐	☐	☐	☐	☐
11.	Assignments are returned in time	☐	☐	☐	☐	☐
12.	Feedbacks on the assignments helped in clarifying the concepts	☐	☐	☐	☐	☐
13.	Project proposals are clearly marked and discussed	☐	☐	☐	☐	☐
14.	Results and grade card of the examination were provided on time	☐	☐	☐	☐	☐
15.	Overall, I am satisfied with the programme	☐	☐	☐	☐	☐
16.	Guidance from the programme coordinator and teachers from the school	☐	☐	☐	☐	☐

After filling this questionnaire send it to:
Programme Coordinator, School of Vocational Education and Training,
Room no. 19, Block no. 1, IGNOU, Maidangarhi, New Delhi- 110068

MPDD-IGNOU/P.O.1T/Feb,2012

ISBN-978-81-266-5891-6