COURSE GUIDE

CYB 213
INFORMATION SECURITY MODELS

Course Team MUOYIBOFARHE Nweke Ozichi PhD – (Course writer)
Dr. Odunola Olanloye **(**Content Editor)

NATIONAL OPEN UNIVERSITY OF NIGERIA

National Open University of Nigeria
Headquarters
University Village
Plot 91, Cadastral Zone
Nnamdi Azikiwe Expressway
Jabi, Abuja

Lagos Office
14/16 Ahmadu Bello Way
Victoria Island, Lagos

E-mail : centralinfo@nou.edu.ng
URL: www.nou.edu.ng

Printed 2024

ISBN: 978-978-786-278-0

CONTENTS

INTRODUCTION

Welcome **to CYB 213 Information Security Models.** CYB 213 is a two-credit unit course that has a minimum duration of one semester. It is a compulsory course for graduate students that are enrolled in BSc Cybersecurity at the National Open University of Nigeria. The course guides you through different advanced and classical security models, access control mechanism.

COURSE COMPETENCIES

- Get familiar with different security models.
- Protect Data at Rest and During Transmission by using different access control mechanisms.
- Protect System and Network Infrastructure

COURSE OBJECTIVES

- To get familiar with different classical security models
- To perform and implement access control list.
- To teach students different access control mechanisms implementation guidelines.

WORKING THROUGH THIS COURSE

To successfully complete this course, read the study units, listen to the audios and videos, do all assessments, open the links and read, participate in discussion forums, read the recommended books and other materials provided, prepare your portfolios, and participate in the online facilitation.

Each study unit has introduction, intended learning outcomes, the main content, conclusion, summary and references/further readings. The introduction will tell you the expectations in the study unit. Read and note the intended learning outcomes (ILOs). The intended learning outcomes tell you what you should be able to do at the completion of each study unit.

So, you can evaluate your learning at the end of each unit to ensure you have achieved the intended learning outcomes. To meet the intended learning outcomes, knowledge is presented in texts, video and links arranged into modules and units. Click on the links as may be directed but where you are reading the text off line, you will have to copy and paste the link address into a browser. You can download the audios and videos to view offline. You can also print or download the texts and save in your computer or external drive. The conclusion gives you the

theme of the knowledge you are taking away from the unit. Unit summaries are presented in downloadable audios and videos.

There are two main forms of assessments – the formative and the summative. The formative assessments will help you monitor your learning. This is presented as in-text questions, discussion forums and Self-Assessment Exercises.

The summative assessments would be used by the university to evaluate your academic performance. This will be given as Computer Base Test (CBT) which serve as continuous assessment and final examinations. A minimum of three computer base test will be given with only one final examination at the end of the semester. You are required to take all the computer base tests and the final examination.

STUDY UNITS

There are 16 study units in this course divided into four modules. The modules and units are presented as follows:

MODULE 1

Unit 1 Bell-La Padula model
Unit 2 Biba model
Unit 3 Clark-Wilson model
Unit 4 Brewer and Nash model
Unit 5 Graham-Denning model
Unit 6 Harrison-Ruzzo-Ullman (HRU)

MODULE 2

Unit 1 Access Control List (ACL)
Unit 2 Mandatory access control (MAC)
Unit 3 Role-based access control (RBAC) resources
Unit 4 Context-based access control (CBAC)
Unit 5 Lattice-based access control (LBAC)

MODULE 3

Unit 1 Capability-based security
Unit 2 Non-interference (security)

MODULE 4

Unit 1 Object-capability model
Unit 2 Take-grant protection model
Unit 3 Protection ring

REFERENCES AND FURTHER READINGS

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015*) "Security in Computing"*, 5th Edition. Prentice Hall.

Bishop, Matt. (2002) *"Computer Security: Art and Science"*, 1st Edition. Addison-Wesley Professional.

Harrison, Michael A.; Ruzzo, Walter L.; Ullman, Jeffrey D. *"Protection in Operating Systems"*, Communications of the ACM, Vol. 19, No. 8, 1976. https://dl.acm.org/doi/10.1145/360303.360333

Gollmann, Dieter. (2011) *"Computer Security"*, 3rd Edition. Wiley.

Gatien Ducornaud (2023). *Role Based Access Control (RBAC) in the context of Smart Grids Implementing and Evaluating a Role Based Access Control System for Configuration Loading in a Substation from a Desktop*. Stockholm, Sweden.

Sandhu, Ravi; Ferraiolo, David; Kuhn, Richard. (2013*) "Role-Based Access Control (1st ed.)"*, NIST/ITL Bulletin.

Ferraiolo, David F.; Kuhn, D. Richard; Chandramouli, Ramaswamy. (2007) *"Role-Based Access Control, Second Edition"*, Artech House.

Sandhu, Ravi S. (1993) *"Lattice-Based Access Control Models"*, IEEE Computer, Vol. 26, No. 11, November 1993. https://ieeexplore.ieee.org/document/241422

Bell, D.E.; LaPadula, L.J. (1973) *"Secure Computer Systems: Mathematical Foundations and Model"*, MITRE Technical Report M74-244. https://apps.dtic.mil/sti/pdfs/AD0770768.pdf

PRESENTATION SCHEDULE

The presentation schedule gives you the important dates for the completion of your computer-based tests, participation in forum discussions and participation at facilitation. Remember, you are to submit all your assignments at the appropriate time. You should guard against delays and plagiarisms in your work. Plagiarism is a criminal offence in academics and is highly penalized.

ASSESSMENT

There are two main forms of assessments in this course that will be scored. The Continuous Assessments and the final examination. The continuous assessment shall be in three-fold. **There will be two Computer Based Assessment. The computer-based assessments will be given in accordance to university academic calendar. The timing must be strictly adhered to**. The Computer Based Assessments shall be scored a maximum of 10% each, while your participation in discussion forums and your portfolio presentation shall be scored maximum of 10% if you meet 75% participation. Therefore, the maximum score for continuous assessment shall be 30% which shall form part of the final grade.

The final examination for CYB 213 will be maximum of two hours and it takes 70 percent of the total course grade. The examination will consist of 70 multiple choice questions that reflect cognitive reasoning.

Note: You will earn 10% score if you meet a minimum of 75% participation in the course forum discussions and in your portfolios otherwise you will lose the 10% in your total score. You will be required to upload your portfolio using google Doc. What are you expected to do in your portfolio? Your portfolio should be note or jottings you made on each study unit and activities. This will include the time you spent on each unit or activity.

HOW TO GET THE MOST FROM THE COURSE

To get the most in this course, you need to have a personal laptop and internet facility. This will give you adequate opportunity to learn anywhere you are in the world. Use the Intended Learning Outcomes (ILOs) to guide your self-study in the course. At the end of every unit, examine yourself with the ILOs and see if you have achieved what you need to achieve.

Carefully work through each unit and make your notes. Join the online real time facilitation as scheduled. Where you missed the scheduled online real time facilitation, go through the recorded facilitation session at your own free time. Each real time facilitation session will be video recorded and posted on the platform.

In addition to the real time facilitation, watch the video and audio recorded summary in each unit. The video/audio summaries are directed to salient part in each unit. You can assess the audio and videos by clicking on the links in the text or through the course page.

Work through all self-assessment exercises. Finally, obey the rules in the class.

FACILITATION

You will receive online facilitation. The facilitation is learner centred. The mode of facilitation shall be asynchronous and synchronous. For the asynchronous facilitation, your facilitator will:

- Present the theme for the week;
- Direct and summarise forum discussions;
- Coordinate activities in the platform;
- Score and grade activities when need be;
- Upload scores into the university recommended platform;
- Support you to learn. In this regard personal mails may be sent.
- Send you videos and audio lectures; and podcast

For the synchronous:

- There will be eight hours of online real time contact in the course. This will be through video conferencing in the Learning Management System. The eight hours shall be of one-hour contact for eight times.
- At the end of each one-hour video conferencing, the video will be uploaded for view at your pace.
- The facilitator will concentrate on main themes that are must know in the course.
- The facilitator is to present the online real time video facilitation time table at the beginning of the course.
- The facilitator will take you through the course guide in the first lecture at the start date of facilitation

Do not hesitate to contact your facilitator. Contact your facilitator if you:

- do not understand any part of the study units or the assignment.
- have difficulty with the self-assessment exercises
- have a question or problem with an assignment or with your tutor's comments on an assignment.

Also, use the contact provided for technical support.
Read all the comments and notes of your facilitator especially on your assignments, participate in the forums and discussions. This gives you opportunity to socialise with others in the programme. You can raise any problem encountered during your study.
To gain the maximum benefit from course facilitation, prepare a list of questions before the discussion session. You will learn a lot from participating actively in the discussions.

Finally, respond to the questionnaire. This will help the university to know your areas of challenges and how to improve on them for the review of the course materials and lectures.

COURSE INFORMATION

Course Code: **CYB213**
Course Title: **Information Security Models**
Credit Unit: 2
Course Status: Compulsory
Course Blub: This course covers various computer, classical and advanced security models and access control mechanisms in computer security. It also covers tools, guides and techniques for implementing these models and access control mechanisms.
Semester: Second
Course Duration: 13 Weeks
Required Hours for Study: 65

ICE BREAKER

You are welcome to CYB 213 Information Security Models, a two-unit course. Please upload your profile such as picture, workplace address, GSM number and other details on your wall. What are your expectations in this course? I am sure you are going to enjoy the course, please fasten your seat belt as you take off. Once again you are welcome.

MAIN COURSE

CONTENT

MODULE 1 CLASSICAL SECURITY MODELS

Introduction

This module delves into classical security models, which are foundational to understanding and implementing robust security policies in computer systems. These models provide structured frameworks to manage and control access to sensitive information, ensuring confidentiality, integrity, and preventing conflicts of interest. The module covers six pivotal models: Bell-LaPadula, Biba, Clark-Wilson, Brewer and Nash, Graham-Denning, and Harrison-Ruzzo-Ullman (HRU). Each model addresses specific security requirements and challenges, offering unique mechanisms to enforce access controls.

Unit 1 Bell-La Padula model
Unit 2 Biba model
Unit 3 Clark-Wilson model
Unit 4 Brewer and Nash model
Unit 5 Graham-Denning model
Unit 6 Harrison-Ruzzo-Ullman (HRU)

In each unit, I will explore a particular topic in detail and highlight self-assessment exercises at the end of the unit. Finally, I highlight resources for further reading at the end of each unit.

UNIT 1 BELL-LA PADULA MODEL

Unit Structure

1.1 Introduction

You will learn from this unit the definition, terminologies and purpose of the bell-LaPadula model. After studying the unit, you will be equipped with skills of the basic concepts of the Bell-Lapadula Model and its distinguishing features.

1.2 Learning Outcomes

By the end of this unit, you will be able to:

- define the Bell–LaPadula model, terminologies.
- identify and classify different security levels in Bell–LaPadula model.

1.3 Bell-La Padula model

1.3.1 Definitions of Bell-La Padula Model

The definitions below cover various aspects of the Bell–LaPadula model, highlighting its conceptual foundations, security properties, mathematical basis, practical applications, and comparison with other security models.

Definition 1: Basic Conceptual Overview
The Bell–LaPadula model is a formal security model designed to maintain the confidentiality of data in computer systems. It achieves this by defining access control rules based on security labels assigned to both users (subjects) and data (objects), ensuring that information does not flow from a higher security level to a lower security level.

Definition 2: Security Properties Emphasis
The Bell–LaPadula model is a state machine model used to enforce data confidentiality policies in multi-level security systems. It is characterized by two primary properties: the Simple Security Property (which prevents subjects from reading data at a higher security level)

and the *-Property (which prevents subjects from writing data to a lower security level).

Definition 3: Practical Implementation Perspective
The Bell–LaPadula model is an access control framework commonly implemented in military and government systems to safeguard sensitive information. By assigning security classifications to both data and users, and by restricting access based on these classifications, the model effectively prevents data leakage and ensures compliance with strict confidentiality requirements.

TERMINOLOGIES
The Bell–LaPadula model involves several key terminologies that are essential to understanding how the model operates. Here are the primary terms associated with the Bell–LaPadula model:

1. **Subject:** A subject is an active entity, usually a user or a process, that requests access to objects. Subjects can perform actions on objects within the constraints of the model's rules.
2. **Object:** An object is a passive entity that contains or receives information. Examples of objects include files, databases, and documents.
3. **Security Labels:** Security labels are markings assigned to both subjects and objects. They consist of:
   Classification Levels: Hierarchical levels such as Top Secret, Secret, Confidential, and Unclassified.
   Categories: Non-hierarchical compartments or groups that further specify access requirements.
4. **Simple Security Property (ss-property):** Also known as the "No Read Up" (NRU) rule, this property stipulates that a subject at a given security level cannot read data at a higher security level. It enforces the principle that subjects should not access information for which they lack clearance.
5. **Star (*) Security Property:** Also known as the "No Write Down" (NWD) rule, this property states that a subject at a given security level cannot write information to a lower security level. This prevents the leakage of sensitive information to less secure levels.
6. **Discretionary Security Property (ds-property):** This property involves discretionary access controls where the owner of an object can determine who is allowed to access it. It supplements the mandatory access controls (Simple Security and *-Properties) by adding a layer of user-defined access restrictions.
7. **Access Modes:** Different types of access actions that subjects can perform on objects:
   Read (R): Accessing the contents of an object.
   Write (W): Modifying the contents of an object.

Execute (X): Running or executing an object.

8. **Security Levels:** The hierarchical levels of security clearance and classification, which define the sensitivity of information and the clearance required to access it.
9. **Secure State**: A state where the only permitted access modes of subjects to objects are in accordance with the security policy.
10. **Trusted Subjects**: Subjects that are not restricted by the Star-property and must be shown to be trustworthy with regard to the security policy.
11. **Tranquillity Principle**: Ensures that subjects cannot change their current security level or drop to a lower level than the highest level they have read so far.
12. **Discretionary Access Control (DAC)**: Gives individual users control over who can access the information they own.
13. **Lattice-Based Access Control:** A mathematical framework used in the Bell–LaPadula model where security levels form a lattice structure, allowing for the definition and enforcement of access control rules based on the relationships between different security levels.
14. **State Machine Model:** The Bell–LaPadula model is represented as a state machine, where the system is defined by a set of states, state transitions, and access control rules that ensure transitions between states do not violate security policies.
15. **Mandatory Access Control (MAC):** A type of access control enforced by the system rather than the user's discretion, based on security labels and predefined rules like the Simple Security and *-Properties.
16. **Information Flow Control:** The mechanism that regulates how information can be transferred between subjects and objects, ensuring that it adheres to the confidentiality rules set by the Bell–LaPadula model.

    Understanding these terminologies is crucial for grasping the principles and functioning of the Bell–LaPadula model, as they form the foundation of its access control mechanisms and security policies.

1.3.2 Purpose of The Bell–Lapadula Model

The Bell–LaPadula model serves several key purposes, primarily focused on maintaining the confidentiality and control of sensitive information within a system. Here are the main purposes itemized:

1. **Ensure Data Confidentiality:** The primary purpose is to protect sensitive information from unauthorized access and disclosure.

2. **Implement Access Control Policies**: To provide a formal framework for defining and enforcing access control policies based on security classifications and clearances.
3. **Prevent Unauthorized Read Access**: To enforce the Simple Security Property (No Read Up), ensuring that subjects cannot read data at higher security levels than their clearance.
4. **Prevent Unauthorized Write Access**: To enforce the Star (*) Security Property (No Write Down), ensuring that subjects cannot write data to lower security levels, thereby preventing information leakage.
5. **Support Mandatory Access Control (MAC)**: To implement a system-enforced access control mechanism that does not rely on user discretion, enhancing overall security by adhering to strict rules.

By fulfilling these purposes, the Bell–LaPadula model contributes significantly to the integrity and security of information systems, particularly in environments where confidentiality is paramount.

SELF-ASSESSMENT EXERCISES 1

1. Define the basic concept of the Bell–LaPadula model.
2. Define a subject in the Bell–LaPadula model.

1.4 Security Levels

The Bell–LaPadula model categorizes information into security levels, typically represented as labels such as "Top Secret," "Secret," "Confidential," and "Unclassified." These security levels play a crucial role in enforcing data confidentiality and access control policies within the model. The model's security levels are designed to control access to classified information and prevent unauthorized access to sensitive data.

1.4.1 Definition and Purpose security levels

Security levels in the Bell–LaPadula model represent a hierarchy of sensitivity and clearance within an information system. Each level indicates the degree of confidentiality required for information and the corresponding clearance needed by users to access that information. The purpose of these security levels is to establish a structured and controlled environment where information flow is regulated to prevent unauthorized access and disclosure.

1.4.2 Classifications of security levels (Top Secret, Secret, Confidential, Unclassified)

The Bell–LaPadula model classifies security levels to control access to information based on its sensitivity and the clearance of users. This classification helps maintain data confidentiality by ensuring that information is accessed and handled appropriately according to its level of sensitivity. Here is a detailed classification of security levels commonly used in the model:

Common Security Levels

1. **Top Secret (TS):** This is the highest level of classification. Information classified as Top Secret is extremely sensitive and its unauthorized disclosure could cause exceptionally grave damage to national security. **Examples**: Strategic military plans, intelligence operations, high-level diplomatic communications.

2. **Secret (S):** Secret information is less sensitive than Top Secret but still critical. Its unauthorized disclosure could cause serious damage to national security. **Examples**: Detailed military plans, certain intelligence reports, sensitive government communications.

3. **Confidential (C):** Confidential information requires protection as its unauthorized disclosure could cause damage to national security. **Examples**: Some military communications, lower-level intelligence data, certain governmental or commercial information.

4. **Unclassified (U):** Unclassified information does not require protection under national security terms but may still be protected under other policies (e.g., privacy, proprietary data). **Examples**: Publicly available information, press releases, general communications.

1.4.3 The Bell–Lapadula Model Security Properties

In the Bell–LaPadula model, subjects and objects are core components, each playing a distinct role in the enforcement of access control policies aimed at maintaining data confidentiality. Understanding their definitions and roles is crucial to grasping how the model operates.

1.4.4 Subject

A subject in the Bell–LaPadula model is an active entity, usually a user or a process, that requests access to objects within a computer system. Subjects can perform operations such as reading, writing, or executing on objects, and their actions are governed by the security policies defined in the model.

Characteristics:

1. **Clearance Level**: Each subject is assigned a security clearance level (e.g., Top Secret, Secret, Confidential, and Unclassified) which determines the highest classification level of information they are authorized to access.
2. **Categories/Compartments**: Subjects may also be associated with specific categories that denote their permissions to access certain types of information within their clearance level.
3. **Active Entity**: Subjects initiate actions and interactions with objects.

Examples:

- A military officer with Secret clearance who needs to access confidential battle plans.
- An application process running on a server that processes sensitive user data.

1.4.5 Object

An object in the Bell–LaPadula model is a passive entity that contains or receives information. Objects are the targets of access requests by subjects and include data structures, files, databases, documents, and other types of stored information.

Characteristics:

1. **Classification Level**: Each object is assigned a security classification level (e.g., Top Secret, Secret, Confidential, and Unclassified) which indicates the sensitivity of the information it contains.
2. **Categories/Compartments**: Objects may also be tagged with specific categories that provide additional constraints on access based on the type of information they hold.
3. **Passive Entity**: Objects do not initiate actions; they are accessed or manipulated by subjects.

Examples:

- A document labelled Top Secret containing classified military intelligence.
- A database table containing confidential financial records.

1.4.6 Access Modes: Read, Write, And Execute

The Bell–LaPadula model defines specific access modes to regulate how subjects interact with objects. These access modes are crucial for enforcing the model's confidentiality policies, ensuring that information is accessed and manipulated in a secure manner. Here are the primary access modes defined in the Bell–LaPadula model:

1. **Read Access (Read):** Allows a subject to view the contents of an object.
- **Simple Security Property (No Read Up)**: A subject can read an object only if the object's classification level is less than or equal to the subject's clearance level.
- **Example**: A user with Secret clearance can read documents classified as Secret, Confidential, and Unclassified, but cannot read Top Secret documents.

2. **Write Access (Write):** Allows a subject to modify or update the contents of an object.
- **Star (*) Security Property (No Write Down)**: A subject can write to an object only if the object's classification level is greater than or equal to the subject's clearance level.
- **Example**: A user with Secret clearance can write to documents classified as Secret and Top Secret, but cannot write to Confidential or Unclassified documents.

3. **Execute Access (Execute):** Allows a subject to execute an object, such as running a program or script.
- **Considerations**: Execute access typically does not involve reading or writing data but may still be subject to security policies to prevent unauthorized actions.
- **Example**: A user might need Top Secret clearance to execute a sensitive program designed for Top Secret operations.

Additional Access Modes (Context-Dependent)
While the Bell–LaPadula model primarily focuses on read and write access, other access modes can be defined based on specific system requirements. These might include:

4. **Append Access (Append):** Allows a subject to add data to an object without being able to modify existing content. **Example**: Logging systems where users can add entries but cannot alter existing log data.

5. **Delete Access (Delete)**: Allows a subject to remove an object or its content. **Considerations**: Deleting sensitive information must be carefully controlled to avoid unauthorized data removal.

6. **Create Access (Create)**: Allows a subject to create new objects within the system. **Example**: A user with appropriate clearance can create new files or documents within a classified information repository.

1.4.7 Interaction Between Subjects and Objects

The interaction between subjects and objects in the Bell–LaPadula model is governed by two main properties to ensure the confidentiality of information:

1.4.8 Simple Security Property (No Read Up)

The Simple Security Property, also known as the "No Read Up" rule, states that a subject at a given security level may not read an object at a higher security level. This means that a subject can only access information at the same security level or lower, ensuring that sensitive information is not accessed by unauthorized individuals.
Example: A user with Secret clearance cannot read a Top-Secret document.

1.4.9 Property (No Write Down)

The * (Star) Security Property, also known as the "No Write Down" rule, states that a subject at a specific classification level cannot write data to a lower classification level. This means that a subject can only write to objects at the same security level or higher, ensuring that sensitive information is not modified or disclosed to unauthorized individuals.
Example: A user with Secret clearance cannot write data to a Confidential document, as this could result in leaking sensitive information to a less secure level.

SELF-ASSESSMENT EXERCISE 2

1. What are security labels?
2. Define Security level in Bell–LaPadula model.
3. What is the purpose of establishing security level?
4. What are the Classifications of security levels?

1.5 Strengths of the Bell–LaPadula Model

1. **Confidentiality:** The Bell–LaPadula model effectively enforces data confidentiality by controlling access to classified information based on security levels and clearance levels.

2. **Simple and Easy to Implement:** The model is straightforward to implement and understand, making it a popular choice for many secure operating systems.

3. **Formal Security Policy:** The Bell–LaPadula model provides a formal security policy that ensures the security of sensitive information by controlling access based on security levels and clearance levels.

4. **Mandatory Access Control (MAC):** The model forms the basis for Mandatory Access Control (MAC) in many secure operating systems, ensuring that access is strictly controlled based on security levels and clearance levels.

5. **Trusted Subjects:** The model allows for trusted subjects that are not restricted by the Star-property, enabling them to perform operations that require higher security levels

1.6 Limitations of the Bell–LaPadula Model

1. **No Clear Distinction between Protection and Security:** The model does not clearly distinguish between protection and security, which can lead to confusion.

2. **No Treatment of Covert Channels:** The model does not extensively address covert channels, which are channels that can be used to pass information without being detected.

3. **No Treatment of Networks of Systems:** The model does not address networks of systems, which can be a significant limitation in modern computing environments.

4. **No Integrity and Availability:** The model primarily focuses on confidentiality and does not address other security aspects like integrity and availability, which may require additional security models and mechanisms.

5. **Tranquillity Principle:** The tranquillity principle ensures that subjects cannot change their current security level or drop to a

lower level than the highest level they have read so far, which can be a limitation in certain scenarios

In summary, the Bell–LaPadula model is a robust security model that effectively enforces data confidentiality and mandatory access control. However, it has limitations in addressing other security aspects like integrity and availability, and does not clearly distinguish between protection and security.

SELF-ASSESSMENT EXERCISE 3

1. Define a Subject.
2. Define an Object.
3. List three different types of access actions that subjects can perform on objects.
4. Why are strategies military plans classified as Top secret?
5. Subjects can perform operations such as _____, _____ or _____on objects.
6. A document and database table are example of _____

1.7 Summary

The Bell-LaPadula model serves a crucial role in ensuring the confidentiality of sensitive information in environments where data classification and security clearances are essential. By enforcing strict access control rules through its simple security property and *-property, the model prevents unauthorized access and leakage of classified information, thereby maintaining data confidentiality and supporting the design of secure systems.

At the end of this unit, you have learnt the definition of the Bell–LaPadula model, the reasons and purposes for implementing the Bell–LaPadula model and some etymologies associated the Bell–LaPadula model. In the next section, I will introduce you to Bida Model.

1.8 References/Further Readings/Web Sources

Bell, D.E.; LaPadula, L.J. "*Secure Computer Systems: Mathematical Foundations and Model"*, MITRE Technical Report M74-244, 1973.

Denning, Dorothy E. "*A Lattice Model of Secure Information Flow"*, Communications of the ACM, Vol. 19, No. 5, May 1976.

Sandhu, Ravi S. "*Lattice-Based Access Control Models"*, IEEE Computer, Vol. 26, No. 11, November 1993.

Bishop, Matt. (2002) *"Computer Security: Art and Science"*, 1st Edition. Addison-Wesley Professional.

Gollmann, Dieter. (2011) *"Computer Security"*, 3rd Edition. Wiley, 2011.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing"*, 5th Edition. Prentice Hall.

Tanenbaum, Andrew S.; Bos, Herbert. (2014) *"Modern Operating Systems"*, 4th Edition. Pearson.

Stallings, William. (2016) *"Cryptography and Network Security: Principles and Practice"*, 7th Edition. Pearson.

1.9 Possible Answers to Self-Assessment Exercises

Answers to SEA 1

1. The Bell–LaPadula model is a formal security model designed to maintain the confidentiality of data in computer systems.
2. A subject in the Bell–LaPadula model is an active entity, usually a user or a process, that requests access to objects within a computer system.

Answers to SEA 2

1. Security labels are markings assigned to both subjects and objects. They consist of classification levels and categories.
2. Security levels in the Bell–LaPadula model represent a hierarchy of sensitivity and clearance within an information system.
3. The purpose of these security levels is to establish a structured and controlled environment where information flow is regulated to prevent unauthorized access and disclosure.
4. Top Secret (TS)
   Secret (S)
   Confidential (C)
   Unclassified (U)

Answers to SEA 3

1. Subject: A subject is an active entity, usually a user or a process, that requests access to objects. Subjects can perform actions on objects within the constraints of the model's rules
2. Object: An object is a passive entity that contains or receives information. Examples of objects include files, databases, and documents.
3. Read (R): Accessing the contents of an object.
   Write (W): Modifying the contents of an object.
   Execute (X): Running or executing an object.
4. Information classified as Top Secret is extremely sensitive and its unauthorized disclosure could cause exceptionally grave damage to national security.
5. Reading, writing, or executing.
6. Object

UNIT 2 BIDA MODEL

Unit Structure

2.1 Introduction

You will learn from this unit fundamentals of data integrity, how important it is in information systems and some common threats. You will also be introduced to the biba models and its history, the key concepts of integrity levels, subjects and objects, the core properties and real-world applications of Bida model

2.2 Learning Outcomes

By the end of this unit, you will be able to:

- know data integrity and threats.
- know the basic concepts of Biba models.

2.3 Bida Model

2.3.1 Introduction to the Biba Model and its History

Data integrity is a fundamental concept in computer security that ensures data remains accurate, consistent, and reliable throughout its entire lifecycle. The Biba Model, developed by Kenneth J. Biba in 1975, is a formal state transition system that addresses data integrity by controlling access to data based on integrity levels.

The Biba Model groups data and subjects (processes) into ordered levels of integrity. The model is designed to prevent subjects from corrupting data at a higher integrity level than the subject, or being corrupted by data from a lower level. The Biba Model defines three main goals for preserving data integrity:

1. Prevent data modification by unauthorized parties
2. Prevent unauthorized data modification by authorized parties
3. Maintain internal and external consistency (i.e., data reflects the real world)

The model is characterized by the phrase "read up, write down", which is the inverse of the Bell-LaPadula model's "read down, write up" for confidentiality. In the Biba Model, users can only create content at or below their own integrity level and view content at or above their own integrity level. The Biba Model defines three main security rules:

1. The Simple Integrity Property: a subject at a given level of integrity must not read data at a lower integrity level (no read down).
2. The * (star) Integrity Property: a subject at a given level of integrity must not write to data at a higher level of integrity (no write up).
3. The Invocation Property: a process from below cannot request higher access; only with subjects at an equal or lower level.

The Biba Model has been implemented in various operating systems, such as FreeBSD, Linux, and XTS-400, to ensure data integrity in secure systems.

2.3.2 Key Concepts: Integrity Levels, Subjects, and Objects

The Biba Model, developed by Kenneth J. Biba in 1975, is a security model that focuses on maintaining data integrity. It is based on the

premise that information should not be corrupted by unauthorized or less trustworthy sources. The key concepts in the Biba Model include integrity levels, subjects, and objects.

2.3.3 Integrity Levels

Integrity levels are a hierarchical structure used to rank the trustworthiness of data and users in a system. They are analogous to security levels in the Bell-LaPadula Model but focus on data integrity rather than confidentiality.

High Integrity Level: Represents highly trustworthy and reliable data or users.
Low Integrity Level: Represents less trustworthy and less reliable data or users.
The primary goal is to prevent data from being contaminated by lower integrity levels.

2.3.4 Subjects

In the Biba Model, subjects are active entities (users or processes) that have the capability to interact with objects (data). Each subject operates at a specific integrity level.

Integrity Level of Subjects: Subjects are assigned an integrity level that reflects their trustworthiness and reliability in handling data.
Rules Governing Subjects: Subjects must adhere to the integrity policies to ensure that data is not corrupted by less trustworthy entities.

2.3.5 Objects

Objects are passive entities (files, databases, records, etc.) that store data. Each object is assigned an integrity level based on the trustworthiness and reliability of the data it contains.

Integrity Level of Objects: Objects are assigned an integrity level that reflects the trustworthiness of the data they hold.
Rules Governing Objects: Objects are protected to ensure that their integrity level is not compromised by interactions with lower-level subjects.

2.4 Core Properties of The Biba Model

The Biba Model is an information security model designed to maintain data integrity. It establishes properties and rules to prevent unauthorized

or less trustworthy sources from corrupting data. Here are the key properties and rules of the Biba Model:

2.4.1 Simple Integrity Property: No Read Down

- **Definition:** A subject at a higher integrity level is not allowed to write to an object at a lower integrity level.
- **Purpose:** This rule prevents more trustworthy subjects from contaminating less trustworthy data.
- **Example:** A senior financial auditor (high integrity) cannot write or modify the raw transaction records (low integrity).

2.4.2 Star (*) Integrity Property: No Write Up

- **Definition:** A subject at a lower integrity level is not allowed to read an object at a higher integrity level.
- **Purpose:** This rule ensures that less trustworthy subjects cannot read or access more trustworthy data, preventing potential corruption of high-integrity information.
- **Example:** A data entry clerk (low integrity) cannot read the audited financial statements (high integrity).

2.4.3 Invocation Property: No Invocation from Lower Levels

- **Definition:** A subject cannot invoke (request services or actions) from a subject at a higher integrity level.
- **Purpose:** This rule prevents less trustworthy subjects from indirectly affecting the operations of more trustworthy subjects.
- **Example:** A junior developer (low integrity) cannot invoke administrative commands on a high-integrity production server.

2.5 Real-World Applications

The Biba Model, designed to maintain data integrity by preventing unauthorized modifications, finds significant real-world applications in securing file systems, databases, and networks. In file systems, it ensures that only authorized users can modify critical files, protecting against data corruption. In databases, the Biba Model controls access to sensitive records, ensuring their accuracy and preventing unauthorized changes. For networks, it safeguards the integrity of transmitted data, ensuring that communications and transactions remain trustworthy and unaltered. These applications are crucial in environments like healthcare, finance, government, and corporate settings, where data integrity is paramount.

2.5.1 Secure File System

Implementing the Biba Model in a secure file system involves integrating its principles to ensure data integrity. This approach is crucial for environments where the accuracy and reliability of data are paramount. Below are some real-world applications and how the Biba Model is applied within secure file systems.

1. ***Healthcare Information Systems***

In healthcare, the integrity of patient records is critical. The Biba Model can be used to ensure that sensitive medical data remains uncorrupted.

- **Application:**
- **Electronic Health Records (EHR):** Implement the Biba Model to control access to patient data.
- **Example:** Doctors (high integrity subjects) can write updates to patient records (high integrity objects), but administrative staff (low integrity subjects) can only read these records.

2. ***Financial Systems***

Financial institutions must ensure the integrity of transaction records and financial statements to prevent fraud and errors.

- **Application:**
- **Transaction Management Systems:** Use the Biba Model to control who can modify financial records.
- **Example:** Auditors (high integrity subjects) can access and finalize financial reports (high integrity objects), while clerks (low integrity subjects) can input transaction data without modifying finalized records.

3. ***Government and Defence***

Government and defence sectors handle highly sensitive information that must remain accurate and secure.

- **Application:**
- **Classified Information Systems:** Implement the Biba Model to maintain the integrity of classified documents and communications.
- **Example:** Senior analysts (high integrity subjects) can write reports (high integrity objects), while support staff (low integrity subjects) can read these reports but cannot modify them.

4. ***Corporate Data Management***

Corporations need to ensure the integrity of critical business data, including intellectual property, strategic plans, and financial data.

- **Application:**
- **Document Management Systems:** Use the Biba Model to protect the integrity of corporate documents.

- **Example:** Executives (high integrity subjects) can update strategic plans (high integrity objects), while regular employees (low integrity subjects) can view but not modify these plans.

5. ***Software Development Environments***

In software development, maintaining the integrity of source code and related documents is essential to prevent errors and security vulnerabilities.

- **Application:**
- **Version Control Systems:** Apply the Biba Model to control access to source code repositories.
- **Example:** Senior developers (high integrity subjects) can commit changes to the main repository (high integrity objects), while junior developers (low integrity subjects) can only commit to feature branches.

2.5.2 Secure Databases

The Biba Model, designed to ensure data integrity by controlling access to prevent unauthorized or less trustworthy sources from corrupting data, can be effectively applied to secure databases. Here are some real-world applications:

1. ***Healthcare Databases***

In healthcare, databases store critical patient information, medical records, and treatment histories. Ensuring the integrity of this data is vital.

- **Application:**
- **Electronic Health Records (EHR):** Implement the Biba Model to control access and modifications to patient data.
- **Example:** Doctors (high integrity subjects) can update patient records (high integrity objects), while administrative staff (low integrity subjects) can only read these records, preventing unauthorized modifications.

2. ***Financial Databases***

Financial institutions maintain databases with sensitive data, such as transaction records, customer information, and financial statements. Integrity is crucial to prevent fraud and errors.

- **Application:**
- **Transaction Processing Systems:** Use the Biba Model to manage who can alter transaction data.
- **Example:** Auditors (high integrity subjects) can finalize transactions (high integrity objects), while clerks (low integrity subjects) can input transaction data without modifying finalized records.

3. ***Government Databases***
Government databases often contain sensitive and classified information that must remain accurate and secure.

- **Application:**
- **Classified Information Systems:** Implement the Biba Model to maintain the integrity of classified documents and communication records.
- **Example:** Senior analysts (high integrity subjects) can write and update classified documents (high integrity objects), while support staff (low integrity subjects) can only read these documents but cannot modify them.

4. ***Corporate Databases***
Corporations need to ensure the integrity of business-critical data, including intellectual property, strategic plans, and financial data.

- **Application:**
- **Document Management Systems:** Use the Biba Model to protect the integrity of corporate documents stored in databases.
- **Example:** Executives (high integrity subjects) can update strategic plans (high integrity objects), while regular employees (low integrity subjects) can view but not modify these plans.

5. ***Educational Databases***
Educational institutions manage databases containing sensitive student information, grades, and academic records.

- **Application:**
- **Student Information Systems:** Implement the Biba Model to ensure the integrity of student records.
- **Example:** Academic staff (high integrity subjects) can update student grades and records (high integrity objects), while students (low integrity subjects) can only view their own records.

2.5.3 Secure Networks

The Biba Model, designed to ensure data integrity by controlling how information flows and is accessed, can be effectively applied to secure networks. Here are some real-world applications:

1. ***Corporate Networks***
In corporate environments, maintaining the integrity of network communications and data is crucial to protect against unauthorized modifications and ensure reliable information flow.

- **Application:**
- **Internal Communication Systems:** Implement the Biba Model to manage the integrity of internal emails, messages, and documents exchanged within the corporate network.

- **Example:** Executives (high integrity subjects) can send and modify strategic directives (high integrity objects), while regular employees (low integrity subjects) can receive and read these directives without altering them.

2. *Healthcare Networks*

Healthcare networks require stringent data integrity controls to protect patient information and ensure the accuracy of medical data exchanged across the network.

- **Application:**
- **Health Information Exchanges (HIE):** Implement the Biba Model to control the integrity of data exchanged between hospitals, clinics, and other healthcare providers.
- **Example:** Doctors (high integrity subjects) can update and share patient records (high integrity objects) across the network, while administrative staff (low integrity subjects) can view but not alter these records.

3. *Government Networks*

Government networks handling sensitive and classified information must maintain high levels of data integrity to protect national security and public interests.

- **Application:**
- **Classified Communication Networks:** Implement the Biba Model to safeguard the integrity of classified information transmitted over government networks.
- **Example:** Intelligence officers (high integrity subjects) can transmit and modify classified reports (high integrity objects), while support staff (low integrity subjects) can only access these reports for operational needs.

4. ***Financial Networks***

Financial institutions must ensure the integrity of transactions and communications to prevent fraud and maintain trust in financial systems.

- **Application:**
- **Interbank Transfer Systems:** Implement the Biba Model to control the integrity of transaction data exchanged between banks.
- **Example:** Auditors (high integrity subjects) can approve and modify transaction records (high integrity objects), while bank tellers (low integrity subjects) can process transactions without altering the records.

5. ***Educational Networks***

Educational institutions manage sensitive student and academic data across their networks, requiring strict integrity controls to ensure accurate and trustworthy information.

- **Application:**
- **Academic Information Systems:** Implement the Biba Model to protect the integrity of student records and academic data transmitted within educational networks.
- **Example:** Faculty members (high integrity subjects) can update student grades and records (high integrity objects), while students (low integrity subjects) can only view their own records without making modifications.

SELF-ASSESSMENT EXERCISEs

1. The Biba model was developed by who and what year?
2. What's the main interest of Biba Model?
3. State the three (3) properties of Biba model and give an example under each.

2.6 Summary

The Biba Model's integrity levels, subjects, and objects work together to maintain data integrity within a system. By enforcing rules that prevent data corruption, the Biba Model ensures that information remains reliable and trustworthy. Understanding these key concepts is crucial for implementing robust data integrity policies in any information system.

In this unit, I explored the basic concepts of Bida model and the key concepts associated with Bida model. I also explored the core properties and real-world applications of bida model. In the next unit, you wil be learning about the concepts of Clark-Wilson Model.

2.7 References/Further Readings/Web Sources

Biba, Kenneth J. *"Integrity Considerations for Secure Computer Systems"*, MITRE Technical Report TR-3153, 1977. https://apps.dtic.mil/sti/pdfs/ADA039324.pdf

Gollmann, Dieter. (2011*) "Computer Security*", 3rd Edition. Wiley.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing"*, 5th Edition. Prentice Hall.

Bishop, Matt. (2002) *"Computer Security: Art and Science"*, 1st Edition. Addison-Wesley Professional.

2.8 Possible Answers to Self-Assessment Exercises

1. Kenneth J. Biba in 1975
2. It addresses data integrity by controlling access to data based on integrity levels.
3. Simple Integrity Property: no read down
   Star (*) Integrity Property: no write up
   Invocation Property: no invocation from lower levels

   i. Simple Integrity Property: no read down
   Example: A senior financial auditor (high integrity) cannot write or modify the raw transaction records (low integrity).

   ii. Star (*) Integrity Property: no write up
   Example: A data entry clerk (low integrity) cannot read the audited financial statements (high integrity).

   iii. Invocation Property: no invocation from lower levels
   Example: A junior developer (low integrity) cannot invoke administrative commands on a high-integrity production server.

UNIT 3 CLARK-WILSON MODEL

Unit Structure

3.1 Introduction

You will learn from this unit the history, components, core principles, advantages, disadvantages and practical applications of the Clark-Wilson model. After studying this unit, you will be equipped with the knowledge and skills for implementing the Clark-Wilson model

3.2 Learning Outcomes

By the end of this unit, you will be able to:

- core principles and concepts of the Clark-Wilson model

3.3 Clark-Wilson Model

3.3.1 Introduction to the Clark-Wilson Model and Its History

The Clark-Wilson model is a security model designed to ensure data integrity in commercial environments by enforcing well-formed transactions and separation of duties. Developed by David D. Clark and David R. Wilson in 1987, this model is particularly effective for systems that manage sensitive and critical data. The following sections shows a

detailed explanation of the key concepts and components of the Clark-Wilson model:

3.3.2 Components of the Clark-Wilson Model

1. **Constrained Data Items (CDIs)**: These are data items that must maintain integrity. The model protects these items through well-formed transactions.
2. **Unconstrained Data Items (UDIs)**: These data items do not require the same level of integrity protection as CDIs.
3. **Integrity Verification Procedures (IVPs)**: These procedures verify that CDIs are in a valid state.
4. **Transformation Procedures (TPs)**: These are well-formed transactions that change the state of CDIs. They must be certified to ensure they maintain integrity.
5. **Authentication and Audit:** The model requires authentication of all users and logging of all modifications to ensure accountability and detect potential security breaches
6. **Certification Rules (CRs)**: These rules ensure that IVPs and TPs maintain the integrity of CDIs. There are specific certification rules:
   i. **CR1**: All TPs must be certified to ensure they transform the system from one valid state to another.
   ii. **CR2**: For all TPs, the security officer must maintain a list of authorized users and the TPs they are allowed to execute.
7. **Enforcement Rules (ERs)**: These rules enforce the execution of TPs and ensure that users only perform authorized actions.

3.3.3 Core Principles in the Clark-Wilson Model

1. **Role Division:** The model mandates that critical tasks be divided into distinct steps, with each step assigned to different roles. This ensures that no single individual has complete control over the entire process, reducing the risk of unauthorized modifications.
2. **Well-formed Transactions:** The concept of well-formed transactions requires that data manipulations be performed in a manner that preserves integrity. By enforcing SoD, the model ensures that these transactions are conducted by multiple individuals, thereby reducing the chances of fraudulent or erroneous data manipulation.
3. **Enforcement through Certification and Enforcement Rules:** These define the constraints and procedures necessary to maintain data integrity. They specify which roles can execute specific operations.

Enforcement Rules (E-rules) also enforces the separation of duties by ensuring that the system adheres to the certification rules, typically through access control mechanisms.

4. **Prevention of Conflicts of Interest:** By separating duties, the Clark-Wilson Model helps prevent conflicts of interest. For example, in financial systems, separating the roles of transaction approval and transaction execution ensures that no single individual can both authorize and execute a payment, thus mitigating the risk of embezzlement.
5. **Monitoring and Auditing:** The model supports continuous monitoring and auditing of transactions and roles. This oversight ensures that the SoD principle is maintained and that any deviations or attempts to circumvent the controls are detected and addressed promptly.
6. **Separation of Duties:** This principle ensures that no single individual has control over all critical aspects of a transaction. It requires different users to perform different roles in a process to prevent fraud and errors.

3.3.4 Practical Example

Consider a payroll system:

Role 1: Employee A can input payroll data (e.g., hours worked, pay rate).

Role 2: Employee B can approve the payroll data entered by Employee A.

Role 3: Employee C can process the payroll after it has been approved by Employee B.

In this scenario, no single employee has the ability to both input and approve payroll data or to process payroll without it being approved. This separation of duties ensures that payroll data integrity is maintained and reduces the risk of fraud.

3.3.5 How the Clark-Wilson Model Works

1. **User Role Assignment**: Users are assigned specific roles that determine which TPs they can execute. This ensures that users cannot perform unauthorized actions.
2. **Certification of Procedures**: All TPs must be certified by security officers to ensure they maintain system integrity. This certification process includes ensuring that TPs can only be executed in a way that preserves the integrity of CDIs.
3. **Logging and Auditing**: The model requires logging of all TP executions to create an audit trail. This helps in detecting and responding to any security breaches.

3.3.6 Advantages of The Clark-Wilson Model

1. **Enhanced Security**: By enforcing well-formed transactions and separation of duties, the model significantly reduces the risk of data corruption and fraud.
2. **Flexibility**: The model can be adapted to different types of commercial systems and is particularly effective in environments where data integrity is critical.
3. **Auditability**: The requirement for logging and auditing all transactions provides a clear audit trail, making it easier to detect and respond to security incidents.

3.3.7 Disadvantages of The Clark-Wilson Model

1. **Complexity**: Implementing the Clark-Wilson model can be complex, especially in large systems with many users and data items.
2. **Performance Overhead**: The need for certification, logging, and auditing can introduce performance overheads.
3. **Dependency on Human Factors**: The effectiveness of the model relies heavily on proper certification of procedures and diligent monitoring by security officers.

SELF-ASSESSMENT EXERCISE

1. What is the main purpose of designing the Clark Wilson model?
2. The Clark Wilson model was developed in what year?
3. What are Constrained Data items (CDI)?
4. Which data items do not require the same level of integrity protections CDIs?
5. What is separation of duty in Clark Wilson model?

3.4 Summary

The Clark-Wilson Model is a fundamental principle designed to maintain data integrity by dividing tasks among multiple roles. This prevents any single user from having unchecked control over critical processes, thereby mitigating risks associated with fraud and errors. The model's structured approach to enforcing SoD through certification and enforcement rules, along with monitoring and auditing, ensures robust data integrity in commercial environments.

In this unit, I explored the basic concepts of Clark-Wilson Model and the key concepts associated with Clark-Wilson Model. I also explored the core properties and real-world applications of Clark-Wilson Model.

In the next unit, you wil be learning about the concepts of Brewer and Nash model.

3.5 References/Further Readings/Web Sources

Clark, David D.; Wilson, David R. *"A Comparison of Commercial and Military Computer Security Policies"*, IEEE Symposium on Security and Privacy, 1987. https://ieeexplore.ieee.org/document/6234

Sandhu, Ravi S. *"Lattice-Based Access Control Models"*, IEEE Computer, Vol. 26, No. 11, November 1993. https://ieeexplore.ieee.org/document/241422

Gollmann, Dieter. (2011*) "Computer Security"*, 3rd Edition. Wiley.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing"*, 5th Edition. Prentice Hall.

Bishop, Matt. (2002) *"Computer Security: Art and Science"*, 1st Edition. Addison-Wesley Professional.

3.6 Possible Answers to Self-Assessment Exercises

1. The Clark-Wilson model is a security model designed to ensure data integrity.
2. Developed by David D. Clark and David R. Wilson in 1987
3. These are data items that must maintain integrity. The model protects these items through well-formed transactions.
4. Unconstrained Data Items (UDIs)
5. This principle ensures that no single individual has control over all critical aspects of a transaction. It requires different users to perform different roles in a process to prevent fraud and errors.

UNIT 4 BREWER AND NASH MODEL

Unit Structure

4.1 Introduction

You will learn from this unit the definition, key concepts of Brewer and Nash Model. After studying the unit, you will be equipped with skills know what Brewer and Nash Model is all about. You will also have the required background knowledge of the history, advantages and disadvantages Brewer and Nash Model.

4.2 Learning Outcomes

By the end of this unit, you will be able to:

- know what Brewer and Nash Model is.
- identify the importance and history Brewer and Nash Model.

4.3 Brewer and Nash Model

4.3.1 Introduction to the Brewer and Nash Model and its history

The Brewer and Nash model, also known as the Chinese Wall model, is a security model designed to prevent conflicts of interest by enforcing

access controls based on the user's previous actions. It was developed by David Brewer and Michael Nash in 1989, specifically to address issues in the financial and consulting industries where conflicts of interest are a significant concern.

4.3.2 Key Concepts of the Brewer and Nash Model

1. **Conflict of Interest Classes (COIs)**: These are groups of datasets that are in conflict with each other. For example, in a financial context, datasets belonging to competing companies would be in the same conflict of interest class.
2. **Company Datasets (CDs)**: These are specific datasets within each conflict-of-interest class. Each dataset contains information about a particular company.
3. **Access Restrictions**: The model restricts access based on the user's previous accesses. If a user has accessed a dataset in a particular conflict of interest class, they are restricted from accessing other datasets in the same class.

4.3.3 Core Principles of the Brewer and Nash Model

1. **No Information Flow Across Conflicts of Interest**: The primary principle is that there should be no flow of information that could lead to a conflict of interest. This means that if a user has accessed information from one dataset within a conflict-of-interest class, they cannot access information from another dataset in the same class.
2. **Dynamic Access Control**: Unlike other models that use static access control lists, the Brewer and Nash model dynamically adjusts access permissions based on the user's activity. This dynamic nature ensures that access decisions are made in real-time to prevent conflicts of interest.

4.3.4 How the Brewer and Nash Model Works

1. **Initial Access**: When a subject (user) first attempts to access a dataset (object), the access is granted if there are no conflicts of interest based on their previous activities.
2. **Conflict Check**: For any subsequent access requests, the system checks the conflict-of-interest classes. If the subject has accessed a dataset in a particular class, they are denied access to other datasets in the same class.
3. **Audit and Monitoring**: The system maintains a history of access attempts and grants to ensure that no conflicts of interest arise. This audit trail helps in reviewing and ensuring compliance with the security policies.

4.3.5 Implementation Challenges

Implementing the Brewer and Nash Model can be complex due to several factors:

1. **Tracking Access History**: The system must continuously track which objects each user has accessed to enforce the access rules dynamically.
2. **Defining COI Classes**: Clearly defining and maintaining the COI classes is essential but can be challenging, especially in large and dynamic environments where business relationships and competitive landscapes frequently change.
3. **Performance Overhead**: Evaluating access control decisions in real-time based on access history can introduce significant performance overhead, particularly in large systems with numerous users and objects.

4.3.6 Example Scenario

Consider a financial consulting firm where consultants provide advice to multiple clients. Clients A and B are competitors and belong to the same conflict of interest class.

1. **Initial Access**: A consultant accesses Client A's dataset. Since this is the first access, it is granted.
2. **Subsequent Access**: The same consultant then tries to access Client B's dataset. The system checks the conflict-of-interest class and denies access because the consultant has already accessed Client A's dataset.

4.3.7 Advantages of the Brewer and Nash Model

1. **Prevents Conflicts of Interest**: By dynamically controlling access based on previous activities, the model effectively prevents conflicts of interest.
2. **Real-time Access Control**: The model's dynamic nature allows for real-time decisions on access, enhancing security and compliance.
3. **Flexibility**: It can be applied to various industries where conflicts of interest are a concern, such as finance, law, and consulting.

4.3.8 Disadvantages of the Brewer and Nash Model

1. **Complex Implementation**: The dynamic nature of the model and the need to maintain access history can make implementation complex.
2. **Performance Overhead**: Continuous monitoring and dynamic decision-making can introduce performance overhead.
3. **Limited Scope**: The model is specifically designed to address conflicts of interest and may not be suitable for other types of security concerns.

SELF-ASSESSMENT EXERCISE

1. What is the main purpose of designing the Brewer and Nash model?
2. The Brewer and Nash model, also known as _____
3. The Brewer and Nash model was developed in what year?
4. What is **Conflict of Interest Classes (COIs)**?
5. Which datasets are specific datasets within each conflict-of-interest class. Each dataset contains information about a particular company.

4.4 Summary

The Brewer and Nash model is an effective security model for environments where conflicts of interest must be strictly managed. By dynamically controlling access based on a user's access history and the conflict-of-interest classes, it ensures that sensitive information is not accessed inappropriately, thus maintaining the integrity and confidentiality of the data.

At the end of this unit, you have learnt the definition, the importance of and the history of the Brewer and Nash model. In the next unit, you will be introduced to the Graham-Denning model.

4.5 References/Further Readings/Web Sources

Brewer, David F.C.; Nash, Michael J. *"The Chinese Wall Security Policy"*, IEEE Symposium on Security and Privacy, 1989. https://ieeexplore.ieee.org/document/36295

Sandhu, Ravi S. *"Lattice-Based Access Control Models"*, IEEE Computer, Vol. 26, No. 11, November 1993. https://ieeexplore.ieee.org/document/241422

Gollmann, Dieter. (2011) *"Computer Security"*, 3rd Edition. Wiley

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing"*, 5th Edition. Prentice Hall.

Bishop, Matt. (2002) *"Computer Security: Art and Science"*, 1st Edition. Addison-Wesley Professional.

4.6 Possible Answers to Self-Assessment Exercises

1. The Brewer and Nash model is a security model designed to prevent conflicts of interest by enforcing access controls based on the user's previous actions.
2. the Chinese Wall model
3. It was developed by David Brewer and Michael Nash in 1989
4. These are groups of datasets that are in conflict with each other.
5. Company Datasets (CDs)

UNIT 5 GRAHAM-DENNING MODEL

Unit Structure

5.1 Introduction

You will learn from this unit fundamentals of the Graham-Denning model and its history, key Concepts, components and operations. You will also be introduced to the security polices, exampl scenario, advantages and disadvantages of Graham-Denning model.

5.2 Learning Outcomes

By the end of this unit, you will be able to:

- Know the basic concepts of Graham-Denning model.

5.3 Graham-Denning Model

5.3.1 Introduction to the Graham-Denning model and its history

The Graham-Denning model is a formal model for computer security that provides a framework for defining and managing the rights and privileges of subjects (users) over objects (resources) in a computer system. This model was introduced by Graham and Denning in 1972

and is fundamental in understanding how to control access and maintain security in a system.

5.3.2 Key Concepts of the Graham-Denning model

1. **Subjects**: Entities (usually users or processes) that can perform actions on objects.
2. **Objects**: Resources (such as files, databases, or devices) on which actions can be performed.
3. **Rights**: Permissions that subjects have over objects, such as read, write, execute, or delete.

5.3.3 Components of the Graham-Denning Model

The model is based on an access control matrix that defines the rights each subject has over each object. The rows of the matrix represent subjects, the columns represent objects, and the cells contain the rights of the corresponding subject over the corresponding object.

5.3.4 Operations of the Graham-Denning model Works

The Graham-Denning model defines a set of eight primitive operations that can be performed to manage and control access rights:

1. **Create Object (CO)**: This operation allows a subject to create a new object and specifies the initial rights associated with that object.
2. **Delete Object (DO)**: This operation allows a subject to delete an existing object from the system, thus removing all associated rights.
3. **Create Subject (CS)**: This operation allows a subject to create a new subject with specified initial rights.
4. **Delete Subject (DS)**: This operation allows a subject to delete an existing subject from the system, thus removing all associated rights.
5. **Read Access Right (r)**: This operation allows a subject to read the rights that another subject has over an object.
6. **Grant Access Right (g)**: This operation allows a subject to grant specific rights to another subject over an object.
7. **Delete Access Right (d)**: This operation allows a subject to delete specific rights of another subject over an object.
8. **Transfer Access Right (t)**: This operation allows a subject to transfer rights they possess to another subject.

5.3.5 Security Policies

The Graham-Denning model supports the implementation of security policies by providing a formal way to define and manage access rights. Policies can be defined to specify who can perform the primitive operations and under what conditions.

5.3.6 Example Scenario

Consider a simple scenario with three subjects (S1, S2, and S3) and two objects (O1, O2):

1. **Initial State**:
   S1 has read (r) and write (w) rights over O1.
   S2 has read (r) rights over O1.
   S3 has no rights over O1 or O2.
   S1 has the right to grant (g) rights over O1.
2. **Grant Operation**: S1 uses the grant operation to give S3 read (r) rights over O1. The access control matrix is updated to reflect this change.
3. **Transfer Operation**: S2 uses the transfer operation to transfer their read (r) rights over O1 to S3. The access control matrix is updated accordingly.
4. **Delete Operation**: S1 uses the delete operation to remove S3's read (r) rights over O1. The access control matrix is updated to reflect this change.

5.3.7 Advantages of the Graham-Denning Model

1. **Granular Control**: Provides detailed and granular control over who can access what resources and what operations they can perform.
2. **Flexibility**: Can be adapted to various types of systems and security requirements.
3. **Formal Framework**: Offers a formalized approach to defining and managing access rights, which is useful for both theoretical analysis and practical implementation.

5.3.8 Disadvantages of the Graham-Denning Model

1. **Complexity**: Managing the access control matrix and the primitive operations can become complex, especially in large systems with many subjects and objects.
2. **Scalability**: The model may face scalability issues as the number of subjects and objects increases, leading to a large and potentially unwieldy access control matrix.

3. **Performance Overhead**: The need to continuously manage and update the access control matrix can introduce performance overhead.

SELF-ASSESSMENT EXERCISE

1. What is the main purpose of designing the Graham-Denning model?
2. The Graham-Denning model was developed in what year?
3. List five (5) operations of Graham-Denning model?

5.4 Summary

The Graham-Denning model is a foundational security model that provides a structured approach to managing access rights in a computer system. By defining a set of primitive operations and using an access control matrix, it offers a formal framework for ensuring that access to resources is controlled and managed in a secure manner. This model is particularly useful in environments where precise and granular control over access rights is required.

At the end of this unit, you have learnt the fundamentals of the Graham-Denning model and its history, key Concepts, components and operations. You have also been introduced to the security polices, exampl scenerio, advantages and disadvantages of Graham-Denning model. In the next unit, you will learn the Harrison-Ruzzo-Ullman (HRU) model.

5.5 References/Further Readings/Web Sources

Graham, Robert M.; Denning, Peter J. *"Protection - Principles and Practice"*, Proceedings of the AFIPS Spring Joint Computer Conference, 1972. https://dl.acm.org/doi/10.1145/1478873.1478879

Bishop, Matt. (2002*) "Computer Security: Art and Science"*, 1st Edition. Addison-Wesley Professional.

Gollmann, Dieter. (2011) *"Computer Security"*, 3rd Edition. Wiley.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing"*, 5th Edition. Prentice Hall.

5.6 Possible Answers to Self-Assessment Exercises

1. A formal model for computer security that provides a framework for defining and managing the rights and privileges of subjects (users) over objects (resources) in a computer system.
2. This model was introduced by Graham and Denning in 1972
3. Create Object, delete object, create subject, delee subject, read access right, grant access right, delete access right, transfer access right.

UNIT 6 HARRISON-RUZZO-ULLMAN (HRU) MODEL

Unit Structure

6.1 Introduction

You will learn from this unit fundamentals of the Harrison-Ruzzo-Ullman (HRU) model and its history, key Concepts, components and primitive operations. You will also be introduced to the security polices, exampl scenerio, advantages and disadvantages of Graham-Denning model.

6.2 Learning Outcomes

By the end of this unit, you will be able to:

- learn the fundamentals of the Harrison-Ruzzo-Ullman (HRU) model and its history.

6.3 Harrison-Ruzzo-Ullman (Hru) Model

6.3.1 Introduction to the Harrison-Ruzzo-Ullman (HRU) model and its history

The Harrison-Ruzzo-Ullman (HRU) model, introduced by Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman in 1976, is a foundational model in the field of computer security. It extends the Graham-Denning model by providing a more detailed framework for understanding and managing access rights in a computer system. The HRU model is particularly noted for its ability to express changes to the access control matrix through a set of operations and for addressing the security implications of such changes.

6.3.2 Key Concepts of the Harrison-Ruzzo-Ullman (HRU) model

1. **Subjects**: Entities that can perform actions (typically users or processes).
2. **Objects**: Resources upon which actions can be performed (files, databases, devices, etc.).
3. **Access Rights**: Permissions that subjects have over objects (read, write, execute, etc.).
4. **Access Control Matrix**: A matrix that defines the rights each subject has over each object, with subjects as rows, objects as columns, and rights in the cells.

6.3.3 Primitive Operations of the Harrison-Ruzzo-Ullman (HRU) model Works

The HRU model defines a set of primitive operations that can be performed on the access control matrix to manage rights. These operations are:

1. **Create Subject**: Adds a new subject to the system.
2. **Delete Subject**: Removes a subject from the system.
3. **Create Object**: Adds a new object to the system.
4. **Delete Object**: Removes an object from the system.
5. **Enter Right**: Grants a specific right to a subject over an object.
6. **Delete Right**: Revokes a specific right from a subject over an object.

Example Operations
To illustrate the primitive operations, consider an access control matrix with subjects S1, S2 and objects O1, O2.

1. **Create Subject (S3)**: Adds a new subject S3.
- Access Control Matrix: Add a new row for S3.
2. **Delete Subject (S1)**: Removes subject S1.
- Access Control Matrix: Remove the row corresponding to S1.
3. **Create Object (O3)**: Adds a new object O3.
- Access Control Matrix: Add a new column for O3.
4. **Delete Object (O2)**: Removes object O2.
- Access Control Matrix: Remove the column corresponding to O2.
5. **Enter Right (S2, O1, read)**: Grants read right to S2 over O1.
- Access Control Matrix: Add "read" to the cell at (S2, O1).
6. **Delete Right (S2, O1, read)**: Revokes read right from S2 over O1.
- Access Control Matrix: Remove "read" from the cell at (S2, O1).

6.3.4 Security Policies

The HRU model supports the enforcement of security policies by specifying the conditions under which each primitive operation can be executed. These policies ensure that only authorized subjects can make changes to the access control matrix.

6.3.5 Decidability and Security

One of the significant contributions of the HRU model is its analysis of the decidability of the safety problem: determining whether a given subject can ever acquire a particular right to an object. The HRU model shows that this problem is undecidable in the general case, meaning there is no algorithm that can determine this for all possible system configurations.

6.3.6 Example Scenario

Consider a system with two subjects (Alice and Bob) and two objects (File1 and File2). The access control matrix may look like this initially:

	File 1	File 2
Alice	Read	Write
Bob		Read

1. Grant Operation: Alice grants Bob read access to File1.
Update: Enter "read" in the cell (Bob, File1).
Resulting Matrix:

	File 1	File 2
Alice	Read	Write
Bob	Read	Read

2. **Revoke Operation**: Alice revokes Bob's read access to File2.
Update: Delete "read" from the cell (Bob, File2).
Resulting Matrix:

	File 1	File 2
Alice	Read	Write
Bob	Read	

6.3.7 Advantages of the HRU Model

1. **Expressiveness**: The HRU model can represent a wide variety of access control scenarios and policies.
2. **Formal Framework**: Provides a rigorous mathematical foundation for understanding access control.
3. **Extensibility**: Can be extended to include additional operations and policies as needed.

6.3.8 Disadvantages of the HRU Model

1. **Undecidability**: The undecidability of the safety problem in the general case can complicate security analysis.
2. **Complexity**: Managing the access control matrix and ensuring consistency can be complex, especially in large systems.
3. **Performance Overhead**: Frequent updates to the access control matrix can introduce performance overhead.

SELF-ASSESSMENT EXERCISE

1. _____ is the permissions that subjects have over objects (read, write, execute, etc.)
2. Harrison-Ruzzo-Ullman (HRU) model was introduced in the year_____
3. Harrison-Ruzzo-Ullman (HRU) model was developed to _____
4. What is the main purpose of "create object" in Harrison-Ruzzo-Ullman (HRU) model?

6.4 Summary

The Harrison-Ruzzo-Ullman model is a comprehensive framework for managing and analyzing access rights in a computer system. Its formal approach to defining and manipulating access control matrices provides a powerful tool for enforcing security policies, though it also introduces challenges related to complexity and decidability.

You have learnt the fundamentals of the Harrison-Ruzzo-Ullman (HRU) model and its history, key Concepts, components and primitive operations. You have also been introduced to the security polices, exampl scenario, advantages and disadvantages of Harrison-Ruzzo-Ullman (HRU) model.

6.5 References/Further Readings/Web Sources

Harrison, Michael A.; Ruzzo, Walter L.; Ullman, Jeffrey D. *"Protection in Operating Systems"*, Communications of the ACM, Vol. 19, No. 8, 1976. https://dl.acm.org/doi/10.1145/360303.360333

Bishop, Matt. (2002) *"Computer Security: Art and Science"*, 1st Edition. Addison-Wesley Professional.

Gollmann, Dieter. (2011) *"Computer Security"*, 3rd Edition. Wiley.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing"*, 5th Edition. Prentice Hall.

6.6 Possible Answer to Self-Assessment Exercises

1. Access right
2. 1976
3. Understand and manage access rights in a computer system
4. Adds a new object to the system.

MODULE 2 ACCESS CONTROL MECHANISMS

Introduction

Access control mechanisms are critical components of information security, ensuring that only authorized users can access specific resources within a system. This module introduces several key access control models, each with its unique approach to managing permissions and ensuring the security of sensitive data. The module covers five pivotal access control mechanisms: Access Control List (ACL), Mandatory access control (MAC), Role-based access control (RBAC), Context-based access control (CBAC) Lattice-based access control (LBAC).

In each unit, I will explore a particular topic in detail and highlight self-assessment exercises at the end of the unit. Finally, I highlight for further reading at the end of each unit.

Unit 1 Access Control Lists (ACL)
Unit 1 Mandatory access control (MAC)
Unit 2 Role-based access control (RBAC) resources
Unit 3 Context-based access control (CBAC)
Unit 4 Lattice-based access control (LBAC)

UNIT 1 ACCESS CONTROL LISTS (ACL)

Unit Structure

1.1 Introduction

You will learn from this unit the definition, terminologies and concepts of Access Control List. After studying the unit, you will be equipped with skills know what access lists are and its importance in controlling network traffic. You will also have the required background knowledge of the history and evolution of ACL.

1.2 Learning Outcomes

By the end of this unit, you will be able to:

- know what ACL is.
- identify various types of ACL
- be familiar with the tools and configuration associated with different ACL.

1.3 Access Control Lists (ACL)

1.3.1 Overview of ACL in network security

Access Control Lists (ACLs) are a critical element in network security, serving as a fundamental apparatus for controlling admittance to network resources. ACLs are used to define and enforce policies that determine which users or devices are permitted or denied entree to specific network resources, based on a number of criteria such as IP addresses, port numbers and protocols.

1.3.2 What is an ACL?

ACLs are usual set of rules applied to network devices (like routers and switches) to regulate the flow of traffic. These rules specify whether packets should be allowed or denied passage through the network, enhancing security by preventing unauthorized access.

The main function of a ACLs is to examine packet headers, make decisions to permit or deny traffic. This control helps in implementing security policies, managing network performance, and ensuring compliance with regulatory requirements.
One of the most common and straightforward uses of access lists is to filter unwanted packets as part of security policies. For instance, access lists can be configured to regulate traffic patterns by permitting only

specific hosts to access web resources on the Internet while restricting others. By using the right combination of access lists, network managers can effectively enforce almost any security policy they design.

Creating access lists is very similar to programming a series of if-then statements—if a given condition is met, then a specific action is taken. If the specific condition isn't met, nothing happens, and the next statement is evaluated. Access-list statements are essentially packet filters that packets are compared against, categorized by, and acted upon accordingly. Once the lists are built, they can be applied to either inbound or outbound traffic on any interface. Applying an access list causes the router to analyze every packet crossing that interface in the specified direction and take the appropriate action.

1.3.3 Access Control List rules

There are 3 significant rules that a packet follows when it is compared with an access list:

i. The packet is compared to each line of the access list in sequential order, starting with the first line and moving through to the subsequent lines in turn.
ii. The packet is compared to the lines of the access list until a match is found. Once it matches a condition on a line of the access list, the packet is processed, and no further comparisons are made.
iii. Each access list implicitly includes a "deny" at the end. This means that if a packet does not match any of the conditions specified in the access list, it will be discarded.

Each of these rules has significant implications for filtering IP packets with access lists. Therefore, it's important to remember that creating effective access lists requires practice.

1.3.4 How ACLs Work:

- ACLs are sequential, meaning the rules are processed in a specific order, starting from the top. The first matching rule determines whether to permit or deny the packet.
- There is an implicit deny at the end of every ACL, so if no rule matches, the packet will be discarded by default.
- ACLs can be applied to either inbound or outbound traffic on an interface. Inbound ACLs process packets before routing, while outbound ACLs process packets after routing.

The configuration of each of the types of ACLs will be discussed extensively in the following section.

1.4 Types of Access Control List

1.4.1 Standard ACLs

Standard IP access lists control network traffic by evaluating the source IP address in each packet. These lists are created using access-list numbers between 1–99 or within the extended range of 1300–1999, as the access-list type is identified by the number. When an access list is generated with one of these numbers, the router recognizes the associated syntax expected for the list. By selecting numbers within these ranges, you're indicating to the router that a standard IP access list is being created, which will only involve source IP address criteria. The example below demonstrates the range of access-list numbers that can be applied to filter network traffic. The available protocols you can filter depend on the IOS version:

```
Seal(config)#access-list ?
  <1-99>          IP standard access list
  <100-199>       IP extended access list
  <1000-1099>     IPX SAP access list
  <1100-1199>     Extended 48-bit MAC address access list
  <1200-1299>     IPX summary address access list
  <1300-1999>     IP standard access list (expanded range)
  <200-299>       Protocol type-code access list
  <2000-2699>     IP extended access list (expanded range)
  <2700-2799>     MPLS access list
  <300-399>       DECnet access list
  <700-799>       48-bit MAC address access list
  <800-899>       IPX standard access list
  <900-999>       IPX extended access list
  dynamic-extended  Extend the dynamic ACL absolute timer
  rate-limit      Simple rate-limit specific access list
```

Here's the syntax for creating a standard IP access list:

```
Seal(config)#access-list 10 ?
  deny    Specify packets to reject
  permit  Specify packets to forward
  remark  Access list entry comment
```

As mentioned, by selecting access-list numbers 1–99 or 1300–1999, you're indicating that you're creating a standard IP access list, which restricts filtering to source IP addresses only. After picking the access-list number, the next step is to determine if you want to allow (permit) or block (deny) traffic. Let's create a deny statement as an example:

```
Seal(config)#access-list 10 deny ?
  Hostname or A.B.C.D  Address to match
  any                  Any source host
  host                 A single host address
```

The next step involves more detail because there are three options:

a. The first option is the **any** parameter, which permit or deny traffic from any source host or network.
b. The second option is to specify an IP address, which can represent a single host or a range of addresses.
c. The last option is to use the **host** command, which is used to specify a single host exclusively.

The **any** command is straightforward—any source address will match the statement, meaning every packet compared to this line will match. The **host** command is similarly simple, as demonstrated here:

```
Seal(config)#access-list 10 deny host ?
  Hostname or A.B.C.D  Host address

Seal(config)#access-list 10 deny host 172.16.30.2
```

This tells the list to deny any packets from host 172.16.30.2. The default parameter is host. So, if you type access-list 10 deny 172.16.30.2, the router assumes you mean host 172.16.30.2 and that’s precisely how it will show in your running-config.

1.4.2 Extended ACLs

Extended ACLs are more flexible and customizable than standard ACLs, allowing administrators to filter network traffic based on additional criteria beyond just the source IP address. Extended ACLs can filter packets based on destination and source IP addresses, type of protocol (TCP, UDP, ICMP, etc.), and source and destination port numbers.

Using an extended access list will be beneficial because it allows us to specify not only source and destination addresses but also the protocol and port number that identify the upper-layer protocol or application. An extended ACL is exactly what we need to effectively grant user's access to a physical LAN while restricting their access to specific hosts and even specific services on those hosts.

Just so you know, you got to use the extended access-list range from 100 to 199. Oh, and the range from 2000 to 2699 is also an option for extended IP access lists. Once you've picked a number from the extended range, you gotta decide what kind of list entry you wanna make. For this example, I'm gonna go with a deny list entry.

```
seal(config)#access-list 110 ?
  deny    Specify packets to reject
  dynamic Specify a DYNAMIC list of PERMITs or DENYs
  permit  Specify packets to forward
  remark  Access list entry comment
```

And once the type of ACL is established, the next thing to do is to select a protocol field entry:

```
seal(config)#access-list 110 deny ?
  <0-255>   An IP protocol number
  ahp       Authentication Header Protocol
  eigrp     Cisco's EIGRP routing protocol
  esp       Encapsulation Security Payload
  gre       Cisco's GRE tunneling
  icmp      Internet Control Message Protocol
  igmp      Internet Gateway Message Protocol
  ip        Any Internet Protocol
  ipinip    IP in IP tunneling
  nos       KA9Q NOS compatible IP over IP tunneling
  ospf      OSPF routing protocol
  pcp       Payload Compression Protocol
  pim       Protocol Independent Multicast
  tcp       Transmission Control Protocol
  udp       User Datagram Protocol
```

1.5 Configuring Extended ACLs:

- The configuration of extended ACLs involves two main steps:

1. Create an extended ACL using the 'access-list' command with a number in the range of 100-199 or 2000-2699.
2. Apply the extended ACL to an interface using the 'ip access-group' command.

- The syntax for creating an extended ACL is: access-list access-list-number permit | deny protocol source-address source-wildcard destination-address destination-wildcard [operator port]

1.5.1 Example Configuration:

- To allow the administrator's workstation (10.0.0.1/24) unrestricted access to a server (192.168.0.1/24) and deny access from the user's workstation (10.0.0.2/24)
  R1(config)#access-list 100 permit ip 10.0.0.1 0.0.0.0 192.168.0.1 0.0.0.0
  R1(config)#access-list 100 deny ip 10.0.0.2 0.0.0.0 192.168.0.1 0.0.0.0
  R1(config)#int f0/0
  R1(config-if)#ip access-group 100 in

- To allow the user's workstation (10.0.0.2/24) to access the web server (192.168.0.1) on port 80 and deny all other traffic
  R1(config)#access-list 100 permit tcp 10.0.0.2 0.0.0.0 192.168.0.1 0.0.0.0 eq 80
  R1(config)#int f0/0
  R1(config-if)#ip access-group 100 in

Here's a table outlining the differences between standard and extended access lists (ACLs):

Feature	Standard Access List	Extended Access List
Number Range	1-99, 1300-1999	100-199, 2000-2699
Filtering Criteria	Source IP address only	Source and destination IP addresses, protocol types, source and destination ports
Granularity	Less granular	More granular
Control Level	Basic access control	Detailed and specific access control
Use Cases	Basic traffic management, simple scenarios	Complex traffic management, specific scenarios
Configuration Complexity	Simpler to configure	More complex to configure
Example Rule	access-list 10 permit	access-list 100 permit tcp

	192.168.1.0 0.0.0.255	any host 192.168.1.10 eq 80
Application	Typically applied close to the destination	Typically applied close to the source
Common Uses	Permitting or denying access from specific subnets or hosts	Filtering specific types of traffic, enforcing security policies for particular protocols and ports
Wildcard Masks	Used for source IP addresses only	Used for both source and destination IP addresses
Protocol Support	IP only	IP, TCP, UDP, ICMP, and other protocols
Named ACLs	Supported	Supported

1. **Future Trends:**

The future of ACLs may involve the integration of machine learning and artificial intelligence to predict and respond to security threats more effectively. These technologies could help in creating adaptive ACLs that automatically adjust rules based on real-time threat intelligence and network behaviour.

In addition, advanced analytics and visibility tools are expected to enhance the management of ACLs, providing deeper insights into network traffic patterns and enabling more proactive security measures.

SELF-ASSESSMENT EXERCISE

1. Define an Access control list.
2. How ACL does enhance network security?
3. What is the main function of ACL?
4. On a network, where can ACL be applied?
5. State three important rules that a packet follows when it's being compared with an access list.
6. What is the configuration range of Standard ACL?
7. What are the criteria used by extended ACL to filter network traffic?

1.6 Summary

In summary, ACLs are essential for defining and enforcing access policies in a file and a network, providing a robust mechanism for securing network traffic and resources. Their strategic implementation and management are crucial for maintaining a secure and efficient network environment.

At the end of this unit, you have learnt the definition of an ACL, the importance of ACL in controlling network traffic and the history and evolution of ACLs. In the next unit, you will be introduced to the different types and components of ACL.

1.7 References/Further Readings/Web Sources

Tanenbaum, Andrew S.; Bos, Herbert. (2014) *"Modern Operating Systems"*, 4th Edition. Pearson.

Stallings, William. (2018) *"Operating Systems: Internals and Design Principles"*, 9th Edition. Pearson.

Garfinkel, Simson; Spafford, Gene. (2003) *"Practical UNIX and Internet Security"*, 3rd Edition. O'Reilly Media.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing"*, 5th Edition. Prentice Hall.

Gollmann, Dieter. (2011) *"Computer Security"*, 3rd Edition. Wiley.

1.8 Possible Answers to Self-Assessment Exercises

1. ACLs are a set of rules applied to network devices (like routers and switches) to regulate the flow of traffic.
2. These rules in ACLs specify whether packets should be allowed or denied passage through the network, enhancing security by preventing unauthorized access.
3. The main function of a ACLs is to examine packet headers, make decisions to permit or deny traffic.
4. Can be applied to either inbound or outbound traffic on any interface.
5. i. The packet is always compared with each line of the access list in sequential order.
   ii. The packet is compared with lines of the access list only until a match is made.
   iii. There is an implicit "deny" at the end of each access list.
6. Standard ACLs are identified by numbers ranging from 1-99 and 1300-1999.
7. They can filter traffic based on multiple criteria including source and destination IP addresses, protocols (such as TCP, UDP, ICMP), and port numbers.

UNIT 2 MANDATORY ACCESS CONTROL (MAC)

Unit Structure

2.1 Introduction

You will learn from this unit the definition, key concepts of Mandatory access control (MAC). After studying the unit, you will be equipped with skills know what Mandatory access control (MAC) is all about. You will also have the required background knowledge of the benefits and limitation of Mandatory access control (MAC).

2.2 Learning Outcomes

By the end of this unit, you will be able to:

- know what Mandatory access control (MAC) is all about.

2.3 Mandatory Access Control (Mac)

2.3.1 Introduction to Mandatory access control (MAC)

Mandatory Access Control (MAC) is a security approach that restricts access to certain assets and resources based on varying authorization levels. It is a centralized access control system that regulates access to resources based on the clearance levels of users and the attributes of objects they seek to access. MAC is known for its high security levels and is often used in government, military, and financial institutions to protect sensitive data and limit the risk of cyberattacks.

1.3.2 Key characteristics of MAC include:

1. **Centralized Control**: The access control policies are defined and enforced centrally by the system administrator, rather than by individual users or resource owners.
2. **Security Labels**: Each user and resource is assigned a security label that defines their security clearance (for users) or classification (for resources). These labels determine access rights.
3. **Policy Enforcement**: The system uses predefined policies to enforce access control decisions, ensuring that users can only access resources for which they have appropriate clearance.
4. **Non-Discretionary**: Unlike Discretionary Access Control (DAC), where resource owners can set access permissions, MAC policies are not subject to user discretion and cannot be modified by end users.
5. **Granularity**: MAC policies can be very granular, allowing for detailed control over who can access what resources under what conditions.

Examples of MAC implementations include:

- **Security-Enhanced Linux (SELinux)**: A Linux kernel security module offers a mechanism for supporting access control security policies, including MAC.
- **Multilevel Security (MLS)**: A system that uses MAC to enforce access controls based on different security levels, often used in military and government contexts.
- **Bell-LaPadula Model**: A formal security model used to enforce access control in military and government applications, focusing on data confidentiality.

What are the basic principles of MAC?

1. The highest privacy and confidentiality of the organization's resources are paramount. No one has default privileges to access or edit someone's data.
2. Access provisioning is centrally administered.
3. All personnel and resource on the network and in the system has security labels alongside their classification and category.

2.3.3 How MAC works.

- The administrator configures access policies and defines security attributes: confidentiality levels and clearances for accessing different projects and types of resources.
- The administrator assigns a set of attributes to each subject (user or resource that accesses data) and object (file, database, port, etc.).
- When a subject attempts to access an object, the operating system examines the subject's security attributes and decides whether access can be granted.
- To obtain access to the object, the user provides their credentials.

When to use MAC

Mandatory Access control model is known for its high security levels and is often used in government, military, and financial institutions to protect sensitive data and limit the risk of cyberattacks. It's mostly used by government organizations, militaries, and law enforcement institutions. Here are some scenarios and contexts where using MAC is advisable:

1. **High-Security Environments:** MAC is suitable for high-security environments where data confidentiality and integrity are critical. Examples include government agencies, military organizations, and financial institutions.

2. **Sensitive Data Protection:** MAC is effective in protecting sensitive data such as classified information, financial records, and patient records.
3. **Industrial Control Systems:** MAC is used in industrial control systems to protect against cyber threats and ensure the security of critical infrastructure.
4. **Shared Servers or Workstations:** MAC is useful in shared computing environments where multiple users access the same resources. It prevents one user from interfering with or accessing the data of another user.
5. **Centralized Management:** MAC is suitable for environments where centralized management is necessary. It allows administrators to manage access control policies uniformly across the organization.
6. **High-Risk Industries:** MAC is used in high-risk industries such as healthcare, finance, and government to protect sensitive data and ensure compliance with regulatory requirements.
7. **Multilevel Security Systems:** MAC is used in multilevel security systems where different users have different clearance levels and access to different resources.

2.3.4 Use Cases

1. **Government and Military:** MAC is widely used in government and military environments to control access to classified or sensitive information.
2. **Financial Data:** At the financial institutions, MAC is used to address highly sensitive customer and financial records.
3. **Healthcare:** In healthcare, MAC is used to control access to electronic health records (EHRs) and safeguard patient privacy.
4. **Industrial Control Systems:** Important industrial systems, such as power plants, water treatment facilities, and transportation systems, depends on MAC to guard against cyber threats.

SELF-ASSESSMENT EXERCISE

1. What is mandatory access control?
2. Mention some areas where Mandatory Access Control (MAC) can be applied.
3. What are the key characteristics of Mandatory Access Control (MAC)?

2.4 Summary

Mandatory Access Control (MAC) is a robust security approach that restricts access to certain assets and resources based on varying authorization levels. It is a centralized access control system that regulates access to resources based on the clearance levels of users and the attributes of objects they seek to access. MAC is suitable for high-security environments, sensitive data protection, industrial control systems, shared servers or workstations, centralized management, high-risk industries, and multilevel security systems. However, it is not suitable for small business settings, consumer applications, low-security environments, and discretionary access control.

You have learnt the fundamentals of Mandatory Access Control (MAC) and key Concepts, components. You have also been introduced to how MAC works and some use case scenarios, benefits and limitations of Mandatory Access Control (MAC). In the next section, I will introduce you to Role-based access control (RBAC).

2.5 References/Further Readings/Web Sources

Bell, D.E.; LaPadula, L.J. (1973) *"Secure Computer Systems: Mathematical Foundations and Model"*, MITRE Technical Report M74-244, 1973.

Sandhu, Ravi S.; Samarati, Pierangela. (1994) *"Access Control: Principles and Practice"*, IEEE Communications Magazine, Vol. 32, No. 9, 1994. https://ieeexplore.ieee.org/document/312842

Bishop, Matt. (2002) *"Computer Security: Art and Science",* 1st Edition. Addison-Wesley Professional.

Gollmann, Dieter. (2011*) "Computer Security"*, 3rd Edition. Wiley.

Pfleeger, Charles P.; (2015) Pfleeger, Shari Lawrence. "*Security in Computing"*, 5th Edition. Prentice Hall.

Alexander Babko https://www.ekransystem.com/en/blog/mac-vs-dac#id-when-to-use-mac

2.6 Possible Answers to Self-Assessment Exercises

1. Mandatory Access Control (MAC) is a security approach that restricts access to certain assets and resources based on varying authorization levels.
2. Government, military, and financial institutions to protect sensitive data and limit the risk of cyberattacks.
3. Centralized Control, Security Labels, Policy Enforcement, Non-Discretionary and Granularity.

UNIT 3 ROLE-BASED ACCESS CONTROL (RBAC)

Unit Structure

3.1 Introduction

In this unit, you will be introduced to definition, key concepts of Role-based access control (RBAC). After studying the unit, you will be furnished with skills to know what Role-based access control (RBAC) is all about. You will also have the required background knowledge of the advantages and disadvantages of Role-based access control (RBAC).

3.2 Learning Outcomes

By the end of this unit, you will be able to:

- know what Role-based access control (RBAC) is.
- identify the advantages and disadvantages of Role-based access control (RBAC).

3.3 Role-based access control (RBAC)

3.3.1 Introduction to Role-based access control (RBAC)

A Role Based Access Control (RBAC) is, as the name implies, an access control technique built on roles. Contrary to MAC or DAC, where the permissions are directly given to users by the administrator or the data's owner, the user is assigned one or more roles, and each role permits for different permissions.

3.3.2 Role-based access control terms and concepts.

Role-Based Access Control (RBAC) involves several key terms and concepts that are essential for understanding and implementing this access control model. Here are the main terms and concepts:

1. **Role**: A collection of permissions that are grouped together based on job functions or responsibilities within an organization. Example: Administrator, Manager, Employee.
2. **User**: An individual who requires access to a system or resources. Users are assigned roles that define their permissions. Example: Alice, Bob.
3. **Permission**: The authorization to perform a specific operation on a resource. Permissions are assigned to roles, not directly to users. Example: Read patient records, Write customer data.
4. **Session**: A mapping between a user and an activated subset of roles that the user can assume during a login session. Example: Alice logs in and activates her "Manager" role for the session.
5. **Role Assignment**: The process of assigning one or more roles to a user. Users inherit the permissions associated with their assigned roles. Example: Assigning the "Doctor" role to Bob.
6. **Role Hierarchy**: A hierarchical structure where roles can inherit permissions from other roles. Higher-level roles include the permissions of their subordinate roles. Example: The "Manager" role inherits permissions from the "Employee" role.
7. **Separation of Duties (SoD)**: A security principle that ensures no single individual has control over all critical aspects of a task. This prevents conflicts of interest and fraud. Example: A user cannot be assigned both the "Approver" and "Requestor" roles to avoid conflicts.
8. **Constraint**: Conditions or rules that restrict role assignments or role activations to enforce security policies. Example: A user cannot activate the "Auditor" role while also activating the "Finance Manager" role.
9. **Role Engineering**: The process of defining and designing roles and permissions based on organizational needs and job functions. Example: Creating a new "Project Manager" role with specific permissions.
10. **Least Privilege**: A security principle that ensures users are granted the minimum permissions necessary to perform their job functions. Example: An intern is given read-only access to certain documents rather than full access.
11. **Static Separation of Duties (SSoD)**: Constraints that prevent conflicting roles from being assigned to the same user at any time. Example: Preventing a user from being assigned both "Cashier" and "Accountant" roles.

12. **Dynamic Separation of Duties (DSoD)**: Constraints that prevent conflicting roles from being activated simultaneously by the same user. Example: A user with both "Developer" and "Tester" roles can only activate one role at a time in a session.
13. **Access Control List (ACL)**: A list of permissions attached to an object specifying which roles or users are allowed to access the object and what operations they can perform. Example: An ACL for a file might specify that the "Admin" role has read/write access, while the "User" role has read-only access.
14. **Attribute-Based Access Control (ABAC)**: An extension or alternative to RBAC where access decisions are based on attributes (such as user attributes, resource attributes, and environmental conditions) rather than predefined roles. Example: Allowing access to a document only if the user's department attribute matches the document's department attribute.

Understanding these terms and concepts is crucial for effectively implementing and managing RBAC within an organization. They provide the foundation for creating a robust and scalable access control system that aligns with organizational policies and security requirements.

3.3.3 Implementing RBAC in operating systems

Implementing Role-Based Access Control (RBAC) in operating systems and applications involves several steps:

1. Define Roles and Permissions:
- Identify the different roles within your organization and define the permissions associated with each role.
- Determine the resources and actions that each role should have access to.

2. Create Groups and Assign Roles:
- Create groups that represent each role and assign the corresponding permissions to those groups.
- Ensure that each user is assigned to the appropriate group based on their role.

3. Configure Access Control Policies: Configure the access control policies to enforce the roles and permissions defined. Ensure that the policies are applied consistently across the organization.

4. Test and Monitor the RBAC Model:
- Test the RBAC model to ensure that it is functioning correctly and that users have the appropriate access to resources.

- Monitor the RBAC model to identify any issues or potential security risks.

5. **Review and Update the RBAC Model:** Regularly review and update the RBAC model to ensure that it remains effective and aligned with the organization's changing needs.

3.3.4 Best Practices for Implementing RBAC

1. **Use a Centralized Management System:** Use a centralized management system to manage roles, permissions, and access control policies. This helps to ensure consistency and reduces administrative overhead.
2. **Use a Hierarchical Role Structure:** Use a hierarchical role structure to simplify role management and reduce the number of roles. This helps to ensure that users have the appropriate access to resources based on their role.
3. **Use Role Inheritance:** Use role inheritance to simplify role management and reduce the number of roles. This helps to ensure that users have the appropriate access to resources based on their role.
4. **Use Role Activation and Expiration:** Use role activation and expiration to manage the lifecycle of roles and ensure that users have the appropriate access to resources. This helps to ensure that users do not have access to resources that they are no longer authorized to access.
5. **Use Auditing and Logging:** Use auditing and logging to track access to resources and identify potential security risks. This helps to ensure that security incidents are detected and responded to promptly.

3.3.5 Tools and Technologies for Implementing RBAC

1. **Active Directory:** Use Active Directory to manage roles, permissions, and access control policies. Active Directory provides a centralized management system for managing roles and permissions.
2. **RBAC Tools:** Use RBAC tools such as Auth0, Okta, or OneLogin to manage roles, permissions, and access control policies. These tools provide a centralized management system for managing roles and permissions.
3. **Scripting and Automation:** Use scripting and automation to simplify role management and reduce administrative overhead. This helps to ensure that roles are managed consistently and efficiently.
4. **Security Information and Event Management (SIEM) Systems:** Use SIEM systems to monitor and analyze security-

related data and identify potential security risks. SIEM systems provide real-time monitoring and analysis of security-related data.

5. **Compliance and Regulatory Requirements:** Ensure that the RBAC model complies with relevant compliance and regulatory requirements. This helps to ensure that the RBAC model is effective and aligned with the organization's changing needs.

SELF-ASSESSMENT EXERCISE

1. What is the main difference between Mandatory Access Control (MAC) or Discretionary Access Control (DAC) and Role Based Access Control (RBAC)?
2. Define role and permission in relation to Role Based Access Control (RBAC).
3. What are the best practices for implementing RBAC?
4. What are the tools and technologies needed for implementing RBAC?

3.4 Summary

Implementing Role-Based Access Control (RBAC) in operating systems and applications involves several steps and best practices. By following these steps and best practices, organizations can ensure that their RBAC model is effective and aligned with their changing needs.

You have learnt the fundamentals of Role-Based Access Control (RBAC) and its key Concepts and components. You have also been introduced to the tools, technology and best practices for implementing RBAC.

3.5 References/Further Readings/Web Sources

Gatien Ducornaud (2023). *Role Based Access Control (RBAC) in the context of Smart Grids Implementing and Evaluating a Role Based Access Control System for Configuration Loading in a Substation from a Desktop*. Stockholm, Sweden.

Sandhu, Ravi; Ferraiolo, David; Kuhn, Richard. (2013) *"Role-Based Access Control (1st ed.)"*, NIST/ITL Bulletin.

Ferraiolo, David F.; Kuhn, D. Richard; Chandramouli, Ramaswamy. (2007) *"Role-Based Access Control, Second Edition"*, Artech House.

Ferraiolo, David F.; Kuhn, D. Richard; Chandramouli, Ramaswamy. (2003) *"Role-Based Access Control"*, Artech House.

3.6 Possible Answers to Self-Assessment Exercises

1. MAC or DAC, where the permissions are directly given to users by the administrator or the data's owner, the user is assigned one or more roles, and each role permits for different permissions.
2. **Role**: A collection of permissions that are grouped together based on job functions or responsibilities within an organization. Example: Administrator, Manager, Employee.
   Permission: The authorization to perform a specific operation on a resource. Permissions are assigned to roles, not directly to users. Example: Read patient records, Write customer data.
3. Use a Centralized Management System
   Use a Hierarchical Role Structure
   Use Role Inheritance
   Use Role Activation and Expiration
   Use Auditing and Logging
4. Active Directory
   RBAC Tools
   Scripting and Automation
   Security Information and Event Management (SIEM) Systems
   Compliance and Regulatory Requirements

UNIT 4 CONTEXT-BASED ACCESS CONTROL (CBAC)

Unit Structure

4.1 Introduction

You will learn from this unit the definition, key concepts of Context-based access control (CBAC). And after learning from the unit, you will be equipped with skills of what Context-based access control (CBAC) is all about. You will also have the required background knowledge of the benefits implantation, best practice and the challenges of Context-based access control (CBAC).

4.2 Learning Outcomes

By the end of this unit, you will be able to:

- know what Context-based access control (CBAC) comprises.
- identify the importance and challenges of Context-based access control (CBAC).

4.3 Context-based access control (CBAC)

4.3.1 Introduction to Context-based access control (CBAC)

Context-Based Access Control (CBAC) is a sophisticated access control model that makes dynamic access decisions based on the context in which a request is made. Unlike traditional models that rely on static

roles or permissions, CBAC evaluates various contextual factors such as time, location, device type, user activity, and other environmental attributes to grant or deny access.

4.3.2 Key Concepts in CBAC

1. **Context:** Context refers to the circumstances or conditions in which access requests occur. It encompasses various attributes, including:
- **Time:** The specific time or time range when access is requested.
- **Location:** The physical or logical location from where the access request originates.
- **Device:** The type or identity of the device used to make the request.
- **User Activity:** The actions or behaviors of the user prior to the access request.
- **Environmental Factors:** Any other conditions or states of the system or environment, such as network security status or the presence of an active threat.

2. **Contextual Attributes:** Attributes are the specific data points used to define the context. These can be classified as:
- **Intrinsic Attributes:** Characteristics inherent to the user or device, such as user roles, device ID, or user identity.
- **Extrinsic Attributes:** External conditions and environmental factors, such as time of day, geolocation, or network status.

3. **Policy Definition:** Policies in CBAC are rules that define how different contextual attributes influence access decisions. These policies are typically dynamic and can adapt to changing conditions.

4. **Decision Engine:** The component that evaluates access requests against contextual policies. It dynamically assesses whether to grant or deny access based on real-time contextual data.

5. **Logging and Auditing:**
- CBAC keeps comprehensive logs of access tries and enforcement activities for audit and compliance reasons.
- Logging capabilities offer visibility into network traffic forms, user actions, and security events, permitting actual monitoring and forensic analysis.

4.3.3 Benefits of CBAC

1. **Dynamic Decision-Making:** Unlike static models, CBAC can adapt to changing conditions, making it more flexible and responsive to emerging threats and varying operational environments.

2. **Enhanced Security:** By considering a broader set of attributes, CBAC can more accurately determine the legitimacy of access requests, thereby reducing the risk of unauthorized access.
3. **Fine-Grained Control:** CBAC provides granulated control over access permissions, permitting for more precise and context-aware enforcement of security policies.
4. **Risk Management:** CBAC can integrate risk assessment into access decisions, adjusting permissions based on the perceived risk level of the context.

4.3.4 Implementing CBAC

1. **Identify Contextual Factors:** Determine which contextual attributes are relevant to your organization's security needs. Common factors include time, location, device type, and user activity.
2. **Define Contextual Policies:** Create policies that specify how access decisions should be made based on the identified contextual factors. Use a policy language or framework that supports dynamic conditions, such as XACML (extensible Access Control Markup Language).
3. **Collect Contextual Data:** Implement mechanisms to collect and aggregate contextual data in real-time. This may involve integrating with existing systems, such as identity management systems, location services, and security information and event management (SIEM) systems.
4. **Develop a Decision Engine:** Build or integrate a decision engine capable of evaluating access requests against the defined policies. Ensure it can process real-time contextual data and make quick decisions.
5. **Implement and Test:** Deploy the CBAC system in a controlled environment to test its functionality and effectiveness. Monitor its performance and make necessary adjustments to policies and data collection mechanisms.

4.3.5 How CBAC Works

CBAC operates by monitoring network traffic and evaluating the contextual information associated with each access request. When a user or application attempts to access a resource, CBAC's policy engine analyses the relevant contextual factors and compares them against the predefined access control policies. Based on this analysis, CBAC makes a dynamic decision to either grant or deny access. If access is granted, CBAC may also apply additional security measures, such as enforcing stronger authentication requirements or limiting the user's privileges. CBAC's dynamic enforcement capabilities allow it to adapt to changing

security conditions in real-time. For example, if CBAC detects suspicious activity or a potential security threat, it can automatically adjust access control policies to mitigate the risk, such as by blocking access from certain locations or devices.

4.3.6 CBAC in Practice

Use Case 1: Healthcare

- **Scenario:** A hospital wants to control access to patient records based on the context of the request.
- **Contextual Factors:**
- ✓ **Time:** Access allowed only during working hours.
- ✓ **Location:** Access allowed only within the hospital premises.
- ✓ **Device:** Access allowed only from hospital-approved devices.
- **Policy:** A doctor can access patient records only during working hours, from within the hospital, and using a hospital-issued device.

Use Case 2: Corporate Environment

- **Scenario:** A company wants to secure access to its internal network and sensitive documents.
- **Contextual Factors:**
- ✓ **User Role:** Different roles have different levels of access.
- ✓ **Location:** Remote access requires multi-factor authentication (MFA).
- ✓ **Device Security:** Access allowed only from devices with updated security patches.
- **Policy:** Employees can access sensitive documents remotely only if they authenticate via MFA and use a device that meets security standards.

4.3.7 Challenges in Implementing CBAC

1. **Complexity:** CBAC systems can become complex due to the large number of contextual attributes and policies that need to be managed.
2. **Real-Time Data Processing:** Efficiently collecting and processing contextual data in real-time requires robust infrastructure and can be challenging.
3. **Policy Management:** Defining and maintaining dynamic policies that accurately reflect organizational needs and security requirements is a continuous process.
4. **Privacy Concerns:** Collecting contextual data, especially those related to location and user behaviour, may raise privacy issues. It is essential to balance security needs with privacy considerations.

SELF-ASSESSMENT EXERCISE

1. What is context-based access control?
2. In Context-Based Access Control (CBAC), what are the attributes of context?
3. What are the challenges associated with implementing CBAC?

4.4 Summary

Context-Based Access Control (CBAC) represents a significant advancement in access control mechanisms, offering dynamic, context-aware decision-making capabilities. By considering a wide range of contextual factors, CBAC can provide more precise and adaptive security controls, enhancing the overall security posture of organizations. Implementing CBAC requires careful planning and robust infrastructure but offers substantial benefits in terms of flexibility, security, and risk management. As technology evolves, CBAC will continue to adapt and integrate with emerging trends, ensuring its relevance in the ever-changing landscape of cybersecurity.

You have learnt the fundamentals of Context-based access control (CBAC) and its key Concepts and components. You have also been introduced to the implementation of Context-based access control (CBAC), how CBAC works and challenges of Context-based access control (CBAC).

4.5 References/Further Readings/Web Sources

Covington, Michael J.; Moyer, Matthew J.; Ahamad, Mustaque. (2000) *"Generalized Role-Based Access Control for Securing Future Applications"*, 23rd National Information Systems Security Conference. https://csrc.nist.gov/CSRC/media/Publications/conference-paper/2000/10/18/proceedings-of-the-23rd-nissc-2000/documents/papers/2000/paper08.pdf

Corradi, Antonio; Montanari, Rebecca; Tibaldi, Daniela. (2004) *"Context-Based Access Control Management in Ubiquitous Environments"*, Proceedings of the 2004 IEEE International Conference on Network. https://ieeexplore.ieee.org/document/1357028

Kulkarni, Deepak; Tripathi, Anand. (2008) *"Context-Aware Role-Based Access Control in Pervasive Computing Systems"*, Proceedings of the 13th ACM Symposium on Access Control Models and Technologies. https://dl.acm.org/doi/10.1145/1377836.1377844

Hu, Vincent C.; Ferraiolo, David; Kuhn, Rick; Schnitzer, Arthur; Sandlin, Kenneth; Miller, Robert; Scarfone, Karen. (2014) *"Guide to Attribute Based Access Control (ABAC) Definition and Considerations"*, NIST Special Publication 800-162. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf

Beimel, Dov; Peleg, Mor. (2005) *"Context-Sensitive Access Control: With or Without Enforcement"*, Proceedings of the 10th ACM Symposium on Access Control Models and Technologies. https://dl.acm.org/doi/10.1145/1063979.1063984

Toninelli, Alessandra; Montanari, Rebecca; Kagal, Lalana; Lassila, Ora. (2006) *"A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments"*, Proceedings of the 5th International Semantic Web Conference. https://link.springer.com/chapter/10.1007/11926078_52

Georgiadis, Christos K.; Mavridis, Ioannis; Pangalos, George; Thomas, Rolf K. *"Flexible Team-Based Access Control Using Contexts"*, Proceedings of the 6th ACM Symposium on Access Control Models and Technologies, 2001. https://dl.acm.org/doi/10.1145/373256.373258

Joshi, James; Bertino, Elisa; Latif, Usman; Ghafoor, Arif. *"A Generalized Temporal Role-Based Access Control Model"*, IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 1, 2005. https://ieeexplore.ieee.org/document/1377836

Bertino, Elisa; Catania, Barbara; Damiani, Maria Luisa; Perlasca, Paolo. *"GEO-RBAC: A Spatially Aware RBAC"*, Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, 2005. https://dl.acm.org/doi/10.1145/1063979.1063981

Covington, Michael J.; Moyer, Matthew J.; Ahamad, Mustaque. *"Contextual Nesting of Roles in Role-Based Access Control"*, Proceedings of the 6th ACM Symposium on Access Control

Models and Technologies, 2001.
https://dl.acm.org/doi/10.1145/373256.373259

4.6 Possible Answers to Self-Assessment Exercises

1. Context-Based Access Control (CBAC) is a sophisticated access control model that makes dynamic access decisions based on the context in which a request is made.
2. Time, Location, Device, User activity and environmental factors
3. Complexity, Real-time data processing, policy management and privacy concerns

UNIT 5 LATTICE-BASED ACCESS CONTROL (LBAC)

Unit Structure

5.1 Introduction

You will learn from this unit the definition, key concepts of Lattice-based access control (LBAC). After studying the unit, you will be equipped with skills know what Lattice-based access control (LBAC) is all about. You will also have the required background knowledge of the advantages and disadvantages of Lattice-based access control (LBAC).

5.2 Learning Outcomes

By the end of this unit, you will be able to:

- know what Lattice-based access control (LBAC) is all about.
- identify the importance and challenges of Lattice-based access control (LBAC).

5.3 Lattice-Based Access Control (LBAC)

5.3.1 Introduction to Lattice-based access control (LBAC)

Lattice-Based Access Control (LBAC) is a sophisticated access control model that structures permissions and access rights using a lattice framework, where security labels are assigned to both users and

resources. In this system, access decisions are determined by comparing the security labels based on predefined dominance relationships within the lattice. The primary goal of LBAC is to enforce strict information flow policies, ensuring that data is only accessible to authorized users based on their security clearance levels and the classification of the information. This model is particularly effective in environments requiring stringent security measures, such as government and military applications, by preventing unauthorized access and potential information leaks.

5.3.2 Key Concepts in LBAC

1. **Security Labels**: Security labels are assigned to both subjects (users) and objects (resources) to indicate their security levels. Labels typically include classifications such as "Confidential," "Secret," and "Top Secret."
2. **Lattice Structure**: A lattice is a mathematical structure that defines a partial ordering of security labels. Each label is a point in the lattice, and the structure determines the hierarchical relationship among these labels.
3. **Dominance Relation**: Dominance ($\geq$) is a key concept where a label A dominates label B if and only if the security level of A is higher than or equal to B. Access decisions are made based on whether the subject's label dominates the object's label.
4. **Meet and Join Operations**: The "meet" (greatest lower bound) and "join" (least upper bound) operations are used to combine security labels. These operations help in determining the effective security level when multiple labels are involved.
5. **Security Classes**: Security classes are specific levels or categories within the lattice. These classes help in organizing and categorizing both data and users based on their sensitivity and clearance levels.

5.3.3 Benefits of LBAC

1. **Enhanced Security**: LBAC provides a robust security framework by enforcing strict access controls based on predefined security labels and dominance relationships. This ensures that only authorized users can access sensitive information, significantly reducing the risk of unauthorized access and data breaches.
2. **Granular Access Control**: The use of a lattice structure allows for fine-grained access control policies. Security labels can be defined at various levels of granularity, enabling precise control over who can access what information.

3. **Prevention of Information Leakage**: By enforcing policies such as "no read up" and "no write down" (in the Bell-LaPadula model) or "no write up" and "no read down" (in the Biba model), LBAC ensures that information does not flow inappropriately, preventing data leaks and maintaining data confidentiality and integrity.
4. **Flexibility and Scalability**: The lattice structure can be easily extended to accommodate new security labels and classes, making LBAC adaptable to evolving security requirements. This flexibility allows organizations to scale their access control policies as their security needs grow.

5.3.4 Implementing LBAC

1. **Define Security Requirements**: Identify sensitive information and determine the necessary levels of confidentiality, integrity, and availability.
2. **Classify Data and Users**: Categorize data based on sensitivity levels and classify users based on their roles and required access levels.
3. **Design the Lattice Structure**: Create security labels and establish dominance relationships among them to define access control policies.
4. **Implement LBAC Policies**: Configure the system with security labels and enforcement mechanisms to apply the defined policies.
5. **Manage and Monitor**: Continuously manage and monitor the LBAC system to ensure compliance and address any security issues that arise.

5.3.5 How LBAC Works

Lattice-Based Access Control (LBAC) works by assigning security labels to both users and resources, which are structured within a mathematical lattice that defines their hierarchical relationships. Access decisions are made based on the dominance relation, where a user's security label must dominate the resource's label for access to be granted. This structure ensures that information flows only in permissible ways, preventing unauthorized access and information leakage. LBAC enforces strict policies such as "no read up" and "no write down" for confidentiality (as in the Bell-LaPadula model) and "no write up" and "no read down" for integrity (as in the Biba model), ensuring robust and consistent access control.

5.3.6 LBAC in Practice

Use Case 1: Government and Military Systems

Scenario: A government agency handles classified information, with data categorized into "Confidential," "Secret," and "Top Secret" levels.

Implementation:

1. **Classification**: All documents are labelled according to their sensitivity level. Employees are assigned security clearances matching these levels.
2. **Access Control**: Using the Bell-LaPadula model, employees can only read documents at or below their clearance level (no read up) and write documents at or above their clearance level (no write down).
3. **Example**: An employee with a "Secret" clearance can access both "Confidential" and "Secret" documents but cannot access "Top Secret" documents. They can create documents labelled "Secret" or "Top Secret," but not "Confidential."

Benefits:

- Prevents unauthorized access to sensitive information.
- Ensures that data is handled according to its classification level, maintaining confidentiality.

Use Case 2: Healthcare Systems

Scenario: A hospital's information system contains patient records that need to be accessed by various healthcare professionals, each with different roles and access needs.

Implementation:

1. **Classification**: Patient records are labeled with sensitivity levels such as "General," "Sensitive," and "Highly Sensitive."
2. **Access Control**: Using the Biba model, healthcare professionals can only read records at or below their integrity level (no read down) and write records at or above their integrity level (no write up).
3. **Example**: A nurse can access "General" and "Sensitive" patient records but cannot access "Highly Sensitive" records. A doctor with higher integrity clearance can access all levels but can only modify records within their clearance level.

Benefits:

- Maintains data integrity by ensuring that only authorized personnel can modify patient records.
- Protects patient privacy by restricting access to sensitive health information.

5.3.7 Challenges in Implementing LBAC

1. **Complexity of Lattice Structure**: Designing and maintaining the lattice structure requires a deep understanding of security requirements and the relationships between different security levels. Organizations must invest time and resources in defining security labels, establishing dominance relationships, and ensuring consistency across the system.
2. **Scalability Issues**: As the organization grows or security needs evolve, scaling the LBAC system can become challenging. Adding new security labels or modifying existing ones may disrupt established access control policies and require careful planning to maintain integrity and security.
3. **Integration with Existing Systems**: Integrating LBAC into existing IT infrastructure, especially legacy systems, can be complex and may require custom development or middleware solutions. Ensuring seamless interaction with other access control mechanisms and applications is crucial to avoid operational disruptions.
4. **Performance Overhead**: Enforcing LBAC policies can introduce additional computational overhead, especially in systems with large numbers of users and resources.

SELF-ASSESSMENT EXERCISE

1. What is Lattice-Based Access Control (LBAC)?
2. What is the primary goal of LBAC?
3. Where is LBAC most effective?
4. What are the benefits of LBAC?
5. How does LBAC works?

.

5.4 Summary

Lattice-based access Control (LBAC) represents a sophisticated approach to access control, leveraging mathematical structures to enforce stringent security policies based on defined security labels and dominance relationships. By organizing access rights into a hierarchical lattice framework, LBAC ensures that sensitive information is protected from unauthorized access and maintains data confidentiality and integrity. Despite its benefits, implementing LBAC poses challenges such as complexity in design and scalability, integration with existing systems, and the need for user training and policy management.

However, with careful planning, robust technical solutions, and ongoing monitoring, LBAC can be effectively deployed to meet stringent security requirements in various high-security environments, including

government agencies, healthcare institutions, and financial organizations, thereby safeguarding critical assets and mitigating risks of unauthorized data breaches.

Lattice-Based Access Control, is a security model that uses a hierarchical lattice structure to enforce access control policies based on security labels and dominance relationships. It ensures that only authorized users can access sensitive information, maintaining confidentiality and integrity. While effective for high-security environments like government and healthcare, LBAC implementation requires careful planning due to its complexity and integration challenges.

5.5 References/Further Readings/Web Sources

Sandhu, Ravi S. (1993) *"Lattice-Based Access Control Models"*, IEEE Computer, Vol. 26, No. 11, November 1993. https://ieeexplore.ieee.org/document/241422

Bell, D.E.; LaPadula, L.J. (1973) *"Secure Computer Systems: Mathematical Foundations and Model"*, MITRE Technical Report M74-244. https://apps.dtic.mil/sti/pdfs/AD0770768.pdf

Denning, Dorothy E. (1976) *"A Lattice Model of Secure Information Flow"*, Communications of the ACM, Vol. 19, No. 5, May 1976. https://dl.acm.org/doi/10.1145/360051.360056

5.6 Possible Answers to Self-Assessment Exercises

1. Lattice-Based Access Control (LBAC) is a sophisticated access control model that structures permissions and access rights using a lattice framework, where security labels are assigned to both users and resources.
2. The primary goal of LBAC is to enforce strict information flow policies, ensuring that data is only accessible to authorized users based on their security clearance levels and the classification of the information.
3. This model is particularly effective in environments requiring stringent security measures, such as government and military applications, by preventing unauthorized access and potential information leaks.
4. Enhace security
   Granulaar access control
   Prevention of information leakage
   Flexibility and scalability
5. Lattice-Based Access Control (LBAC) works by assigning security labels to both users and resources, which are structured within a mathematical lattice that defines their hierarchical relationships.

MODULE 3 ADVANCED SECURITY MODELS

Introduction

In today's digital landscape, security is a critical concern for organizations and individuals alike. The constant evolution of cyber threats necessitates the development and implementation of sophisticated security models to protect sensitive information and ensure the integrity of systems. This module, "Advanced Security Models," delves into the theoretical foundations and practical applications of cutting-edge security frameworks designed to address contemporary cybersecurity challenges.

In each unit, I will explore a particular topic in detail and highlight self-assessment exercises at the end of the unit. Finally, I highlight resources for further reading at the end of each unit.

Unit 1 Capability-based security
Unit 2 Non-interference (security)

UNIT 1 CAPABILITY-BASED SECURITY

Unit Structure

1.1 Introduction

You will learn from this unit the definition, terminologies and associated with capability-based security. After studying the unit, you will be equipped with skills of the basic concepts of Capability-based security and its distinguishing features.

1.2 Learning Outcomes

By the end of this unit, you will be able to:

- define the Capability-based security, terminologies.
- implement Capability-based security and identify challenges associated.

1.3 Capability-Based Security

1.3.1 Definition of Capability-Based Security

Capability-based security is an approach to security in computer systems that controls access to objects (such as files or devices) through capabilities. A capability is an unforgeable token or key that grants the holder permission to perform specific operations on an object. This model contrasts with traditional access control models like access control lists (ACLs), which rely on maintaining a central list of permissions associated with each object.

1.3.2 Key Characteristics of Capability-Based Security include

1. **Capabilities:** A capability is a communicable, unforgeable token of authority. Capabilities are often represented as data structures that specify the operations that can be performed on an object, along with a reference to the object itself.
2. **Unforgeability:** Ensuring that capabilities cannot be fabricated or altered by unauthorized entities is crucial. This is typically achieved using cryptographic techniques or by embedding capabilities in a secure hardware environment.
3. **Delegation:** Capabilities can be passed from one entity to another, allowing for flexible and fine-grained delegation of authority. This is analogous to handing someone a key that grants access to a specific resource.
4. **Revocation:** Unlike traditional models, revocation in a capability system can be more complex. Techniques like revocation lists, indirection (using proxy capabilities), or time-limited capabilities are used to manage this.
5. **Least Privilege:** Capability systems naturally support the principle of least privilege, where entities are granted only the minimum set of capabilities necessary to perform their functions.

1.3.3 System Architecture and Design Considerations

1. **Capability Table**: A core component of capability-based systems is the capability table, which maps subjects to their respective capabilities. Each entry in the table represents a capability that a subject possesses.
2. **Capability Registers**: Some systems utilize hardware support for capabilities, storing them in dedicated capability registers. This approach enhances performance and security by leveraging specialized hardware features.
3. **Capability-Based Operating Systems**: Several operating systems have been designed around the capability-based security model. Notable examples include EROS and CapROS.

- ✓ **EROS (Extremely Reliable Operating System)**: EROS is designed to provide high reliability and security using capability-based security. It offers fine-grained access control and efficient management of capabilities.
- ✓ **CapROS**: An evolution of EROS, CapROS improves scalability and performance while maintaining the core principles of capability-based security.

1.3.4 Implementation Challenges and Solutions

1. **Performance Overhead**: Managing capabilities can introduce performance overhead, particularly in systems with high interaction rates. Optimizing capability storage and verification processes is essential.
2. **Compatibility**: Integrating capability-based security with existing systems and applications can be challenging. Solutions often involve hybrid models or middleware to bridge the gap.
3. **User Management**: Ensuring that users understand and correctly manage their capabilities is crucial. User interfaces and tools must be designed to simplify capability management.

1.3.5 Advantages of Capability-based Security

1. **Fine-grained Access Control:** Capabilities allow for very specific permissions, enhancing the precision of access control.
2. **Decentralized Management:** Since capabilities can be managed and distributed without a central authority, they support decentralized systems and scalable security architectures.
3. **Principle of Least Privilege:** The model inherently supports least privilege by making it easier to grant the minimum necessary permissions.

4. **Delegation:** Capabilities make it straightforward to delegate access rights, which is useful in distributed systems and complex organizational structures.
5. **Simplified Access Control:** With capabilities, access checks are simplified. The possession of a capability token directly implies permission, removing the need for complex permission checks at runtime.

1.3.6 Challenges and Limitations

1. **Revocation:** Efficiently revoking capabilities once they have been distributed can be challenging, especially in large, distributed systems.
2. **Distribution:** Securely distributing capabilities and ensuring their integrity and confidentiality over untrusted networks is a critical challenge.
3. **Capability Proliferation:** Managing a large number of capabilities, ensuring they are not lost or leaked, and tracking their distribution can become complex.
4. **Compatibility:** Integrating capability-based security with existing systems and software that use traditional access control models can be difficult.

1.3.7 Real-world Implementations

1. Operating Systems: Systems like KeyKOS, EROS, and CapROS have been designed with capability-based security at their core.
2. Programming Languages: Some languages, such as E and Caja, incorporate capabilities to manage authority and permissions at the language level.
3. Web Browsers and Applications: Modern web browsers, like Google Chrome, use capability-based security principles in their sandboxing mechanisms to isolate processes and restrict permissions.
4. Distributed Systems: Capability-based security is particularly well-suited for distributed systems and microservices architectures where decentralized control and fine-grained permissions are beneficial.

SELF-ASSESSMENT EXERCISE

1. Define Capability-based security.
2. What is capability?

1.4 Summary

Capability-based security offers a robust and flexible model for access control, emphasizing fine-grained permissions, decentralized management, and the principle of least privilege. While it presents certain challenges, particularly around revocation and capability management, its advantages make it a compelling choice for many modern computing environments, especially those requiring high levels of security and scalability.

At the end of this unit, you have learnt the definition of Capability-based security, the Key characteristics, System Architecture and Design Considerations, Advantages and challenges of Capability-based security. In the next section, I will introduce you to Non-interference (security).

1.5 References/Further Readings/Web Sources

Miller, Mark S. (2006) *"Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control"*, Ph.D.

Thesis, Johns Hopkins University. Maffeis, Sergio; Mitchell, John C.; Taly, Ankur. (2010) *"Object Capabilities and Isolation of Untrusted Web Applications"*, IEEE Symposium on Security and Privacy. https://ieeexplore.ieee.org/document/5504804

Barth, Adam; Weinberger, Joel; Song, Dawn (2009) *"Cross-Origin JavaScript Capability Leaks: Detection, Exploitation, and Defense"*, USENIX Security Symposium.https://www.usenix.org/legacy/event/sec09/tech/full_papers/barth.pdf

Blokdyk, Gerardus. (2018*) "Capability-Based Security: The Ultimate Step-By-Step Guide"*. https://www.waterstones.com/book/capability-based-security-the-ultimate-step-by-step-guide/gerardus-blokdyk/9780655157601

Blokdyk, Gerardus. *"Capability Based Security a Complete Guide - 2020 Edition"*. https://www.amazon.com/Capability-Based-Security-Complete-Guide/dp/1867332450

Klein, Gerwin; Andronick, June; Elphinstone, Kevin; Murray, Toby; Sewell, Thomas; Kolanski, Rafal; Heiser, Gernot. (2014)

"Comprehensive Formal Verification of an OS Microkernel", ACM Transactions on Computer Systems, Volume 32, Number 1. https://ts.data61.csiro.au/publications/nicta_full_text/7449.pdf

Devriese, Dominique; Birkedal, Lars; Piessens, Frank. (2016) *"Reasoning About Object Capabilities with Logical Relations and Effect Parametricity"*, 1st IEEE European Symposium on Security and Privacy. https://lirias.kuleuven.be/retrieve/355325

1.6 Possible Answers to Self-Assessment Exercises

1. Capability-based security is an approach to security in computer systems that controls access to objects (such as files or devices) through capabilities.
2. A capability is an unforgeable token or key that grants the holder permission to perform specific operations on an object.

UNIT 2 NON-INTERFERENCE (SECURITY)

Unit Structure

2.1 Introduction

You will learn from this unit the definition, terminologies and associated with Non-interference (security). After studying the unit, you will be equipped with skills of the basic concepts of Non-interference (security) and its distinguishing features.

2.2 Learning Outcomes

By the end of this unit, you will be able to:

- define the Non-interference (security) and the key terminologies.
- implement Non-interference (security) and identify challenges associated.

2.3 Non-interference (security)

2.3.1 Definition of Non-interference (security)

Non-interference is a concept in computer security and formal methods, which ensures that actions at a higher security level do not affect what can be observed at a lower security level. This principle is crucial in preventing information leaks from sensitive or classified operations to less secure or unclassified operations. Non-interference provides a

formal foundation for creating secure systems, particularly in environments where confidentiality and integrity are paramount.

2.3.2 Key Concepts in Non-interference

1. **High-Level and Low-Level Security Domains**:
- **High-Level**: Represents sensitive or classified information and operations.
- **Low-Level**: Represents less sensitive or unclassified information and operations.
2. **Observability**: Non-interference ensures that low-level entities cannot observe any changes or effects resulting from high-level operations.
3. **Formal Definition**: A system is non-interfering if, for any two executions that are identical from the perspective of low-level users, the observable behavior of these users remains identical even if high-level actions differ.
4. **Security Policy**: Non-interference is a stringent security policy that often complements other policies, such as the Bell-LaPadula model for confidentiality and the Biba model for integrity.

2.3.3 Mathematical Formalization

Non-interference is often formalized using mathematical models. Consider a system with states and actions, where actions can be performed by subjects at different security levels:

- **States**: Represent the configuration of the system at any point in time.
- **Actions**: Operations performed by subjects that can change the state of the system.
- **Observables**: Aspects of the state that are visible to subjects at various security levels.

Formally, non-interference can be expressed as follows:
Let S be the set of system states.
Let A be the set of actions.
Let L and H denote low and high security levels, respectively.
Let α: S$\rightarrow$ Observables be a function that maps states to what is observable at the low level.

A system is non-interfering if, for any two sequences of actions $\sigma 1$ and $\sigma 2$ that are indistinguishable to low-level users, and any initial states: $\alpha(\text{exec}(s,\sigma 1))=\alpha(\text{exec}(s,\sigma 2))$ where $\text{exec}(s,\sigma)$ denotes the state resulting from executing the sequence σ starting from states.

2.3.4 Advantages of Non-interference

1. **Strong Confidentiality Guarantees**: Ensures that high-level information does not leak to low-level observers, providing strong confidentiality guarantees.
2. **Formal Verification**: Non-interference can be rigorously verified using formal methods, providing a mathematically sound basis for security assurance.
3. **Simplicity in Policy Enforcement**: The principle provides a clear and straightforward policy for separating different security levels, reducing the risk of inadvertent information leaks.

2.3.5 Challenges and Limitations

1. **Strictness**: Non-interference is a very strict policy and can be difficult to enforce in practical systems, especially where some level of information flow is necessary for functionality.
2. **Performance Overhead**: Implementing non-interference can introduce significant performance overhead, as it may require extensive checking and isolation mechanisms.
3. **Usability**: The rigidity of non-interference can sometimes impede usability and flexibility, as it restricts interactions between different security levels.
4. **Granularity**: Determining the appropriate granularity for observables and actions can be challenging, affecting the practical implementation of non-interference.

2.3.6 Real-world Applications

1. **Military and Government Systems**: Non-interference is often applied in highly secure military and government systems to prevent leaks of classified information.
2. **Secure Operating Systems**: Operating systems designed with strong security guarantees, such as certain versions of SELinux, incorporate principles of non-interference to ensure strict separation between processes of different security levels.
3. **Formal Methods and Verification Tools**: Formal methods and verification tools, such as model checkers and theorem provers, use non-interference to prove the security properties of systems, particularly in critical applications like aerospace and medical devices.

SELF-ASSESSMENT EXERCISE

1. In mathematical formulation of Non-interference (security), what are the different security levels?
2. What are the real-world applications of Non-interference (security)?

2.4 Summary

Non-interference is a foundational principle in computer security that ensures high-level actions do not influence low-level observations, thereby preventing unauthorized information flow. While it provides strong confidentiality guarantees and a robust basis for formal verification, its strictness and potential impact on system performance and usability present significant challenges. Despite these challenges, non-interference remains a crucial concept in the design and analysis of secure systems, particularly in environments where confidentiality and data integrity are critical.

In this unit, I explored the basic concepts of Non-interference (security) and the key concepts associated with Non-interference (security). I also explored the core properties and real-world applications of Non-interference (security).

2.5 References/Further Readings/Web Sources

Goguen, Joseph A.; Meseguer, José. (1982) *"Security Policies and Security Models"*, IEEE Symposium on Security and Privacy.

Sutherland, Dana. (1986) *"A Model of Information"*, Proceedings of the 9th National Computer Security Conference.

Giacobazzi, Roberto; Mastroeni, Isabella. (2004) *"Abstract Non-Interference: Parameterizing Non-Interference by Abstract Interpretation"*, Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages.

Barthe, Gilles; Crespo, Juan Manuel; Kunz, César. (2011*) "Relational Verification Using Product Programs"*, International Conference on Formal Methods, 2011.

Sabelfeld, Andrei; Myers, Andrew C. (2003) *"Language-Based Information-Flow Security"*, IEEE Journal on Selected Areas in Communications, Vol. 21, No. 1.

2.6 Possible Answers to Self-Assessment Exercises

1. States, actions and observables.
2. Military, secure operating systems and Formal Methods and Verification Tools

MODULE 4 COMPUTER SECURITY MODELS AND MECHANISMS

Introduction

In today's interconnected digital world, ensuring the security of computer systems and data is paramount. Computer security models and mechanisms provide the foundational framework for protecting information from unauthorized access, modification, and destruction. These models are designed to enforce access controls, maintain confidentiality, ensure data integrity, and mitigate risks from cyber threats. By understanding various security models and the mechanisms they employ, professionals can effectively design, implement, and maintain secure systems across diverse domains, from government and military operations to financial institutions, healthcare systems, and beyond.

This module explores key concepts such as Object-capability model, Take-grant protection model, Protection ring offering insights into their theoretical foundations, practical applications, and real-world examples. Through this exploration, learners will gain a comprehensive understanding of how these models safeguard sensitive information and contribute to robust cybersecurity practices in the modern digital landscape.

In each unit, I will explore a particular topic in detail and highlight self-assessment exercises at the end of the unit. Finally, I highlight for further reading at the end of each unit.

Unit 1 Object-capability model
Unit 2 Take-grant protection model
Unit 3 Protection ring

UNIT 1 OBJECT-CAPABILITY MODEL

Unit Structure

1.1 Introduction
1.2 Learning Outcomes
1.3 Object-Capability Model
    1.3.1 Definition of Object-capability model
    1.3.2 Key Concepts in the Object-capability model
    1.3.3 Advantages of the Object-capability model
    1.3.4 Challenges and Limitations of Object-capability model
    1.3.5 Real-world implementations

1.1 Introduction

You will learn from this unit the definition, terminologies and associated with Object-capability model. After studying the unit, you will be equipped with skills of the basic concepts of Object-capability model and its distinguishing features.

1.2 Learning Outcomes

By the end of this unit, you will be able to:

- define the Object-capability model and the key terminologies.
- implement Object-capability model and identify challenges associated.

1.3 Object-Capability Model

1.3.1 Definition of Object-capability model

The object-capability model is a security architecture that integrates object-oriented programming principles with capability-based security mechanisms. It focuses on controlling access to objects and their methods through capabilities, which are unforgeable tokens that grant specific rights. This model emphasizes fine-grained access control, encapsulation, and the principle of least authority, making it a powerful approach for designing secure software systems.

1.3.2 Key Concepts in the Object-Capability Model

1. Capabilities:

- Definition: Capabilities are tokens that combine a reference to an object with the authority to perform specific operations on that object.
- Unforgeability: Capabilities are unforgeable, meaning they cannot be created or altered by unauthorized entities. This is typically enforced by the language runtime or the operating system.

2. Objects and Methods:
- Encapsulation: Objects encapsulate state and behavior, exposing methods that can be invoked using capabilities.
- Reference Passing: Capabilities are passed by reference, allowing objects to grant other objects specific access rights.

3. Principle of Least Authority (POLA): Objects should be granted only the minimum capabilities necessary to perform their tasks, reducing the risk of unintended actions and enhancing security.
4. Delegation: Objects can delegate their capabilities to other objects, allowing for flexible and dynamic access control. Delegation is akin to handing over a key that grants specific access.
5. Revocation: While capabilities themselves are not inherently revocable, various patterns and techniques (like revocation lists or intermediary objects) can be used to implement revocation.

1.3.3 Advantages of the Object-Capability Model

1. **Fine-grained Access Control**: Capabilities provide precise control over what operations can be performed on which objects, enhancing security by limiting the scope of authority.
2. **Encapsulation and Modularity**: The model promotes strong encapsulation and modular design, making it easier to reason about and manage complex systems.
3. **Least Privilege**: The principle of least authority reduces the risk of security breaches by ensuring that components operate with the minimal necessary permissions.
4. **Dynamic Access Control**: Capabilities can be dynamically created, distributed, and revoked, allowing for flexible and adaptive security policies.
5. **Simplicity and Clarity**: The explicit use of capabilities simplifies the reasoning about access control and security policies, making it easier to audit and verify systems.

1.3.4 Challenges and Limitations

1. **Revocation**: Implementing effective revocation mechanisms for capabilities can be complex, especially in large and dynamic systems.
2. **Capability Proliferation**: Managing a large number of capabilities and ensuring they are not lost or misused can become challenging.
3. **Integration with Existing Systems**: Integrating the object-capability model with existing systems that use traditional access control mechanisms can be difficult.

4. **Usability**: Designing systems that effectively use capabilities without becoming overly complex or cumbersome for developers requires careful planning and expertise.

1.3.5 Real-world Implementations

1. **Programming Languages**: Languages like E, Cap'n Proto, and Joe-E are designed with the object-capability model in mind, providing built-in support for capabilities and secure access control.
2. **Operating Systems**: Systems like KeyKOS, EROS, CapROS, and seL4 implement capability-based security at the operating system level, providing fine-grained control over system resources.
3. **Web Browsers and Applications**: Web browsers like Google Chrome use principles of the object-capability model in their sandboxing mechanisms to isolate processes and restrict access to resources.
4. **Distributed Systems**: Object-capability principles are well-suited for distributed systems and microservices architectures, where decentralized control and fine-grained permissions are beneficial.

1.3.6 Key Design Patterns and Practices

1. **Caretaker Pattern**: A caretaker object holds capabilities and delegates them to other objects as needed, providing a centralized point of control.
2. **Facet Pattern**: An object exposes different facets, each with a distinct set of capabilities, allowing for controlled access to different aspects of the object's functionality.
3. **Forwarder Pattern**: A forwarder object forwards requests to another object, potentially mediating or restricting access based on capabilities.
4. **Revoker Pattern**: A revoker object manages the revocation of capabilities, providing mechanisms to invalidate or restrict access when necessary.

SELF-ASSESSMENT EXERCISE

1. What is the main focus of object-capability model?
2. Why does objects delegate their capabilities to other objects?
3. What does delegation involves>

1.4 Summary

The object-capability model offers a robust and flexible approach to security, integrating the principles of object-oriented programming with capability-based access control. By emphasizing fine-grained permissions, encapsulation, and the principle of least authority, it provides a powerful framework for designing secure systems. Despite challenges like revocation and capability management, the model's advantages in terms of security, modularity, and clarity make it an attractive choice for many modern computing environments, particularly those requiring high levels of security and adaptability.

In this unit, I explored the basic concepts of Object-capability model and the key concepts associated with Object-capability model. I also explored the core properties and real-world applications of Object-capability model.

1.5 References/Further Readings/Web Sources

Miller, Mark Samuel. (2006) *"Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control",* Ph.D. Thesis, Johns Hopkins University.

Bishop, Matt. (2002) *"Computer Security: Art and Science",* 1st Edition. Addison-Wesley Professional.

Gollmann, Dieter. (2011*) "Computer Security",* 3rd Edition. Wiley.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing",* 5th Edition. Prentice Hall.

Stallings, William. (2016) *"Cryptography and Network Security: Principles and Practice",* 7th Edition. Pearson.

Tanenbaum, Andrew S.; Bos, Herbert. (2014) *"Modern Operating Systems",* 4th Edition. Pearson.

Maffeis, Sergio; Mitchell, John C.; Taly, Ankur. (2010) *"Object Capabilities and Isolation of Untrusted Web Applications",* IEEE Symposium on Security and Privacy.

Barth, Adam; Weinberger, Joel; Song, Dawn. (2009) *"Cross-Origin JavaScript Capability Leaks: Detection, Exploitation, and Defense"*, USENIX Security Symposium.

1.6 Possible Answer to Self-Assessment Exercises

1. It focuses on controlling access to objects and their methods through capabilities, which are unforgeable tokens that grant specific rights.
2. It allows for flexible and dynamic access control.

UNIT 2 TAKE-GRANT PROTECTION MODEL

Unit Structure

2.1 Introduction

You will learn from this unit the definition, terminologies associated with Take-grant protection model. After studying the unit, you will be equipped with skills of the basic concepts of Take-grant protection model and its distinguishing features.

2.2 Learning Outcomes

By the end of this unit, you will be able to:

- define the Take-grant protection model and the key terminologies.
- implement Take-grant protection model and identify challenges associated.

2.3 Take-grant protection model

2.3.1 Definition of the Take-grant protection model

The take-grant protection model is a formal model used to analyze the security properties of access control systems, particularly focusing on how rights (permissions) can be transferred within a system. This model uses a directed graph to represent the access rights between subjects (active entities like users or processes) and objects (passive entities like

files or resources). The main operations in the model are "take" and "grant," which define how rights can be acquired and transferred.

2.3.2 Key Concepts of the Take-Grant Protection Model

1. **Graph Representation**: The system is represented as a directed graph, where nodes represent subjects and objects. Edges between nodes represent the rights that a subject has over an object. For example, an edge from node A to node B with a label "r" indicates that A has read rights over B.
2. **Rights**:

- **Take**: Allows a subject to take rights over an object from another subject. This operation enables the transfer of rights from one node to another.
- **Grant**: Allows a subject to grant its rights over an object to another subject. This operation enables a subject to delegate its permissions to another subject.

3. **Rules and Operations**:

- **Take Rule**: If a subject A has the take right over a subject B, then A can acquire any rights that B has over another object or subject.
- **Grant Rule**: If a subject A has the grant right over a subject B, then A can grant any of its rights over an object to B.
- **Create Rule**: A subject can create a new object and initially has all rights over this object.
- **Remove Rule**: A subject can remove any of its own rights over an object.

2.3.3 Advantages of the Take-Grant Protection Model

1. **Simplicity**: The model is relatively simple and easy to understand, making it suitable for theoretical analysis and teaching.
2. **Formal Analysis**: The take-grant model provides a formal framework for analyzing the distribution and propagation of access rights, helping to identify potential security issues.
3. **Flexibility**: It can represent a wide range of access control scenarios and policies, from simple discretionary access control (DAC) to more complex scenarios.
4. **Visualization**: The graph representation aids in visualizing the structure of access rights and the relationships between subjects and objects, making it easier to understand and manage access control policies.

2.3.4 Challenges and Limitations

1. **Expressiveness**: The model's simplicity comes at the cost of expressiveness. It may not capture all the nuances and complexities of real-world access control systems, such as mandatory access control (MAC) or role-based access control (RBAC).
2. **Scalability**: For large systems with many subjects and objects, the graph can become complex and difficult to manage.
3. **Static Nature**: The model does not inherently account for dynamic changes in the system, such as the creation and deletion of subjects and objects or changes in the security policy.
4. **Lack of Fine-Grained Control**: The basic take and grant operations may not provide sufficient granularity for complex access control requirements, such as conditions based on time or context.

2.3.5 Real-world Applications

1. **Theoretical Analysis**: The take-grant model is primarily used in theoretical contexts to analyze and understand the propagation of access rights and to prove properties about security policies.
2. **Teaching**: It is a useful tool for teaching fundamental concepts of access control and security models in academic settings.
3. **Design of Access Control Systems**: While not directly implemented in real-world systems, the principles of the take-grant model can inform the design and analysis of access control mechanisms.

2.3.6 Example Scenario

Consider a system with three subjects (A, B, and C) and two objects (X and Y). The initial rights are as follows:
A has take rights over B.
B has grant rights over C.
A has read rights over X.
B has write rights over Y.

Operations and Changes in the Graph

1. **A takes rights from B**:
- Since A has take rights over B, A can acquire B's write rights over Y.
- Now, A has read rights over X and write rights over Y.
2. **B grants rights to C**:
- Since B has grant rights over C, B can grant its write rights over Y to C.
- Now, C has write rights over Y.
3. **A grants rights to B**:

- If A has the grant right over B, A can grant its read rights over X to B.
- Now, B has read rights over X and write rights over Y.

Analysis
The model allows us to track how rights propagate through the system and to ensure that certain security properties are maintained. For example, we can verify whether it is possible for C to eventually gain read rights over X, given the initial configuration and the rules of the model.

SELF-ASSESSMENT EXERCISE

1. What is the function of directed graph intake-grant protection model?
2. List the two main operations in take-grant protection model.

2.4 Summary

The take-grant protection model provides a formal framework for understanding and analyzing access control in computer systems. It emphasizes the propagation of rights through simple operations, making it useful for theoretical analysis and teaching. While it has limitations in expressiveness and scalability, the model's simplicity and clarity offer valuable insights into the design and management of access control policies. By representing access rights as a directed graph, the model aids in visualizing and reasoning about the security of systems, contributing to the broader field of computer security and access control.

In this unit, I explained the basic concepts of Take-grant protection model and the key concepts associated with Take-grant protection model. I also explored the core properties and real-world applications of Take-grant protection model.

2.5 References/Further Readings/Web Sources

Bishop, Matt. (2002) *"Computer Security: Art and Science",* 1st Edition. Addison-Wesley Professional.

Gollmann, Dieter. (2011*) "Computer Security",* 3rd Edition. Wiley.

Pfleeger, Charles P.; Pfleeger, Shari Lawrence. (2015) *"Security in Computing",* 5th Edition. Prentice Hall.

Stallings, William. (2016) *"Cryptography and Network Security: Principles and Practice"*, 7th Edition. Pearson.

Tanenbaum, Andrew S.; Bos, Herbert. (2014) *"Modern Operating Systems"*, 4th Edition. Pearson.

2.6 Possible Answers to Self-Assessment Exercises

1. This model uses a directed graph to represent the access rights between subjects (active entities like users or processes) and objects (passive entities like files or resources).
2. The main operations in the model are "take" and "grant," which define how rights can be acquired and transferred

UNIT 3 PROTECTION RINGS

Unit Structure

3.1 Introduction

You will learn from this unit the definition, terminologies and associated with Take-grant protection model. After studying the unit, you will be equipped with skills of the basic concepts of Take-grant protection model and its distinguishing features.

3.2 Learning Outcomes

By the end of this unit, you will be able to:

- define the Take-grant protection model and the key terminologies.
- implement Take-grant protection model and identify challenges associated.

3.3 Protection Rings

3.3.1 Introduction to Protection rings

Protection rings are a key concept in computer security and operating system design, providing a structured approach to privilege levels within a computer system. These rings create layers of security, each with different levels of access and control over system resources. The primary goal of protection rings is to enhance system stability and

security by restricting direct access to critical system functions and data to only the most trusted parts of the system.

3.3.2 Key Concepts of Protection Rings

1. **Ring Structure**:
- **Hierarchy**: Protection rings are organized in a hierarchical manner, typically numbered from 0 to n. Lower-numbered rings have more privileges, while higher-numbered rings have fewer.
- **Typical Ring Assignments**:
  - **Ring 0**: Kernel mode or supervisor mode, with the highest privileges, able to execute all instructions and access all memory.
  - **Ring 1 and 2**: Often used for device drivers and other privileged but less critical services.
  - **Ring 3**: User mode, with the least privileges, where application software runs.
2. **Privilege Levels**: The system enforces different privilege levels, restricting the types of operations that can be performed by code executing at each level. For example, Ring 0 code can perform hardware I/O operations, while Ring 3 code cannot.
3. **Memory Protection**: Protection rings enforce memory protection by controlling access to memory regions based on the current privilege level. This prevents user-mode applications from accessing or modifying kernel memory.
4. **CPU Modes**: Modern CPUs support different operating modes corresponding to the protection rings. Instructions executed in lower-numbered rings can perform more powerful operations compared to those in higher-numbered rings.
5. **System Calls**: User-mode applications interact with the kernel through system calls, which are controlled entry points to the higher-privilege Ring 0. This mechanism ensures that user-mode code cannot directly execute privileged instructions.

3.3.3 Advantages of Protection Rings

1. **Enhanced Security**: By isolating critical system components and restricting access to sensitive operations, protection rings significantly reduce the risk of system compromise due to malicious or faulty software.
2. **System Stability**: Fault isolation prevents bugs and crashes in user-mode applications from affecting the kernel or other critical parts of the system, thereby improving overall system stability.
3. **Controlled Access**: Protection rings provide a structured way to manage access to system resources, ensuring that only trusted code can perform sensitive operations.

4. **Defense in Depth**: The ring model adds multiple layers of defense, making it more difficult for attackers to escalate privileges and compromise the system.

3.3.4 Challenges and Limitations

1. **Complexity**: implementing and managing multiple protection rings can add complexity to the operating system and software development process.
2. **Performance Overhead**: The need to switch between different privilege levels, such as when making system calls, can introduce performance overhead.
3. **Limited Use in Modern Systems**: Many modern operating systems, such as Linux and Windows, primarily use two main privilege levels (kernel mode and user mode), simplifying the model but potentially sacrificing some of the granularity of traditional protection rings.
4. **Compatibility**: Ensuring compatibility and seamless interaction between different layers and their respective privilege levels can be challenging, particularly in complex systems with diverse hardware and software components.

3.3.5 Real-world Implementations

1. **X86 Architecture**: The x86 architecture supports four protection rings (0-3), but most operating systems primarily use Ring 0 for the kernel and Ring 3 for user-mode applications.
2. **Operating Systems**:

- **Windows**: Uses a simplified ring model with two main levels: user mode (Ring 3) and kernel mode (Ring 0).
- **Linux**: Similarly uses user mode and kernel mode, corresponding to Ring 3 and Ring 0, respectively.
- **Multics**: One of the earliest systems to implement the full protection ring model, influencing subsequent operating system designs.

3. **Virtualization**: Modern virtualization technologies leverage protection rings to isolate virtual machines from the host system and from each other. Hypervisors, such as those used in VMware and Hyper-V, operate at an even lower level than the traditional Ring 0, sometimes referred to as Ring -1.

SELF-ASSESSMENT EXERCISE

1. What is protection ring?
2. How does protection ring works?
3. What is the primary goal of protection ring?

3.4 Summary

Protection rings provide a structured and hierarchical approach to managing access control and privileges in computer systems. By isolating critical system components and enforcing strict access controls, protection rings enhance security, stability, and robustness. While the traditional multi-ring model has evolved in modern operating systems to focus on two primary levels, the underlying principles continue to influence system design, particularly in areas like virtualization and secure computing environments. Despite challenges related to complexity and performance, protection rings remain a fundamental concept in the architecture of secure and reliable computing systems.

In this unit, I explained the basic concepts of Protection rings and the key concepts associated with Protection rings. I also explored the core properties and real-world applications of Protection rings.

3.5 References/Further Readings/Web Sources

Silberschatz, Abraham; Galvin, Peter Baer; Gagne, Greg. (2018) *"Operating System Concepts"*, 10th Edition. Wiley.

Hennessy, John L.; Patterson, David A. (2017) *"Computer Architecture: A Quantitative Approach"*, 6th Edition. Morgan Kaufmann.

Tanenbaum, Andrew S.; Bos, Herbert. (2014) *"Modern Operating Systems"*, 4th Edition. Pearson.

Stallings, William; Brown, Lawrie. (2018) *"Computer Security: Principles and Practice"*, 4th Edition. Pearson.

McKusick, Marshall Kirk; Neville-Neil, George V.; Watson, Robert N.M. (2014) *"The Design and Implementation of the FreeBSD Operating System"*, 2nd Edition. Addison-Wesley Professional.

3.6 Possible Answers to Self-Assessment Exercise

1. Protection rings are a key concept in computer security and operating system design, providing a structured approach to privilege levels within a computer system.
2. These rings create layers of security, each with different levels of access and control over system resources.
3. The primary goal of protection rings is to enhance system stability and security by restricting direct access to critical system functions and data to only the most trusted parts of the system.