DATABASE SECURITY

UNIT-1

INTRODUCTION:

- **INTRODUCTION TO DATA BASE SECURITY PROBLEM IN DATABASE SECURITY CONTROLS CONCLUSIONS:**

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

At its core, database security strives to insure that only authenticated users perform authorized activities at authorized times. While database security incorporates a wide array of security topics, notwithstanding, physical security, network security, encryption and authentication, this paper focuses on the concepts and mechanisms particular to securing data. Within that context, database security encompasses three constructs: confidentiality or protection of data from unauthorized disclosure, integrity or prevention from unauthorized data access, and availability or the identification of and recovery from hardware and software errors or malicious activity resulting in the denial of data availability.

In the computing discipline curricula, database security is often included as a topic in an introductory database or introductory computer security course. This paper presents a set of sub-topics that might be included in a database security component of such a course. Mapping to the three constructs of data security, these topics include access control, application access, vulnerability, inference, and auditing mechanisms. Access control is the process by which rights and privileges are assigned to users and database objects. Application access addresses the need to assign appropriate access rights to external applications requiring a database connection. Vulnerability refers to weaknesses that allow malicious users to exploit resources. Inference refers to the use of legitimate data to infer unknown information without having rights to directly retrieve that information. Database auditing tracks database access and user activity providing a way to identify breaches that have occurred so that corrective action might be taken. As the knowledge base related to database security continues to grow, so do the challenges of effectively conveying the material. This paper addresses those challenges by incorporating a set of interactive software modules into each sub-topic. These modules are part of an animated database courseware project designed to support the teaching of database concepts. The courseware covers.

INTRODUCTION

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

The Verizon Business Risk Team, who have been reporting data breach statistics since 2004, examined 90 breaches during the 2008 calendar year. They reported that more than 285 million records had been compromised, a number exceeding the combined total from all prior years of study (Baker et al., 2009). Their findings provide insight into who commits these acts and how they occur. Consistently, they have found that most data breaches originate from external sources, with 75% of the incidents coming from outside the organization as compared to 20% coming from inside. They also report that 91% of the compromised records were linked to organized criminal groups. Further, they cite that the majority of breaches result from hacking and malware often facilitated by errors committed by the victim, i.e., the database owner. Unauthorized access and SQL injection were found to be the two most common forms of hacking, an interesting finding given that both of these exploits are well known and often preventable. Given the increasing number of beaches to database systems, there is a corresponding need to increase awareness of how to properly protect and monitor database systems.

At its core, database security strives to insure that only authenticated users perform authorized activities at authorized times. It includes the system, processes, and procedures that protect a database from unintended activity. The Defence Information Systems Agency of the US Department of Defence (2004), in its Database Security Technical Implementation Guide, states that database security should provide “controlled, protected access to the contents of your database and,in the process, preserve the integrity, consistency, and overall quality of your data” (p. 9). The goal is simple, the path to achieving the goal, a bit more complex. Traditionally database security focused on user authentication and managing user privileges to database objects (Guimaraes, 2006).

This has proven to be inadequate given the growing number of successful database hacking incidents and the increase in the number of organizations reporting loss of sensitive data. A more comprehensive view of database security is needed, and it is becoming imperative for students in the computing disciplines to develop an understanding of the issues and challenges related to database security and to identify possible solutions.

Database security is often included as a topic in an introductory database course or introductory computer security course. However as the knowledge base related to database security continues to grow, so do the challenges of effectively conveying the material. Further, many topics related to database security are complex and require students to engage in active learning to fully comprehend the fundamental nature of database security issues. This paper presents a set of subtopics for inclusion in a database security component of a course. These sub-topics are illustrated using a set of interactive software modules.

As part of a National Science Foundation Course, Curriculum and Laboratory Improvement Grant (#0717707), a set of interactive software modules, referred to as Animated Database Courseware (ADbC) has been developed to support the teaching of database concepts. ADbC consists of over 100 animations and tutorials categorized into four main modules (Database Design, Structured Query Language [SQL], Transactions and Security) and several sub modules. Interactive instructional materials such as animations can often be incorporated into the instructional process to enhance and enrich the standard presentation of important concepts. Animations have been found to increase student motivation, and visualizations have been found to help students develop understanding of abstract concepts which are otherwise considered to be 'invisible' (Steinke, Huk, & Floto, 2003). Further, software animations can be effective at reinforcing topics introduced in the classroom as they provide a venue for practice and feedback.

Specifically, the Security module and corresponding sub-modules will be covered in this paper. These sub-modules cover six areas: access control, row level security, application security as portrayed in a security matrix, SQL injections, database inference, and database auditing.

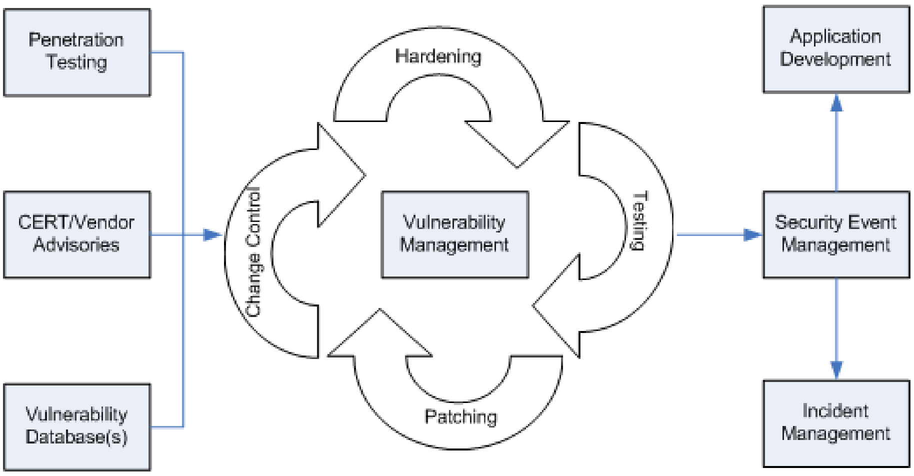
Database Security Topics:

The following presents an organizational structure for presenting database security concepts in a course in which database security is one of many topics. As such the focus is limited and material introductory. While database security incorporates a wide array of security topics, not withstanding, physical security, network security, encryption and authentication, this paper focuses on the concepts and mechanisms particular to securing data. Database security is built upon a framework encompassing three constructs: confidentiality, integrity and availability (Bertino & Sandhu, 2005). Confidentiality or secrecy refers to the protection of data against unauthorized disclosure, integrity refers to the prevention of unauthorized and improper data modification, and availability refers to the prevention and recovery from hardware and software errors as well as from malicious data access resulting in the denial of data availability (Bertino, Byun & Kamra, 2007).

Mapping to these three constructs, a database security component in any course needs to cover access control, application access, vulnerability, inference, and auditing mechanisms. The primary method used to protect data is limiting access to the data. This can be done through authentication, authorization, and access control. These three mechanisms are distinctly different but usually used in combination with a focus on access control for granularity in assigning rights to specific objects and users. For instance, most database systems use some form of authentication, such as username and password, to restrict access to the system. Further, most users are authorized or assigned defined privileges to specific resources. Access control further refines the process by assigning rights and privileges to specific data objects and data sets. Within a database, these objects usually include tables, views, rows, and columns. For instance, Student A may be given login rights to the University database with authorization privileges of a student user which include read-only privileges for the Course_ Listing data table. Through this granular level of access control, students may be given the ability to browse course offerings but not to peruse grades assigned to their classmates. Many students, today, inherently understand the need for granularity in granting access when framed in terms of granting ‘friends’ access to their Facebook site. Limiting access to database objects can be demonstrated through the Grant/Revoke access control mechanism.

DATABASE VULNERABILITY

A Vulnerability Database is a platform aimed at collecting, maintaining, and disseminating information about discovered vulnerabilities targeting real computer systems. Currently, there are many vulnerabilities databases that have been widely used to collect data from different sources on software vulnerabilities (e.g., bugs). These data essentially include the description of the discovered vulnerability, its exploitability, its potential impact, and the workaround to be applied over the vulnerable system. Examples of web-based vulnerabilities databases are the National Vulnerability Database and the Open Source Vulnerability Database.



Security breaches are an increasing phenomenon. As more and more databases are made accessible via the Internet and web-based applications, their exposure to security threats will rise. The objective is to reduce susceptibility to these threats. Perhaps the most publicized database application vulnerability has been the SQL injection. SQL injections provide excellent examples for discussing security as they embody one of the most important database security issues, risks inherent to non-validated user input. SQL injections can happen when SQL statements are dynamically created using user input. The threat occurs when users enter malicious code that 'tricks' the database into executing unintended commands. The vulnerability occurs primarily because of the features of the SQL language that allow such things as embedding comments using double hyphens (- -), concatenating SQL statements separated by semicolons, and the ability to query metadata from database data dictionaries. The solution to stopping an SQL injection is input validation. A common example depicts what might occur when a login process is employed on a web page that validates a username and password against data retained in a relational database. The web page provides input forms for user entry of text data. The user-supplied text is used to dynamically create a SQL statement to search the database for matching records. The intention is that valid username and password combinations would be authenticated and the user permitted access to the system. Invalid username and passwords would not be authenticated. However, if a disingenuous user enters malicious text, they could, in essence, gain access to data to which they have no privilege. For instance, the following string, ' OR 1=1 -- entered into the username textbox gains access to the system without having to know either a valid username or password. This hack works because the application generates a dynamic query that is formed by concatenating fixed strings with the values entered by the user.

For example, the model SQL code might be:

SELECT Count(*) FROM UsersTable

WHERE UserName = ‘contents of username textbox’

AND Password = ‘contents of password textbox’;

When a user enters a valid username, such as ‘Mary’ and a password of ‘qwerty’, the SQL query

becomes:

SELECT Count(*) FROM UsersTable

WHERE UserName=‘Mary’

AND Password=‘qwerty’;

However, if a user enters the following as a username: ‘OR 1=1 -- the SQL query becomes:

SELECT Count(*) FROM UsersTable

WHERE UserName=‘‘ OR 1=1 - -’

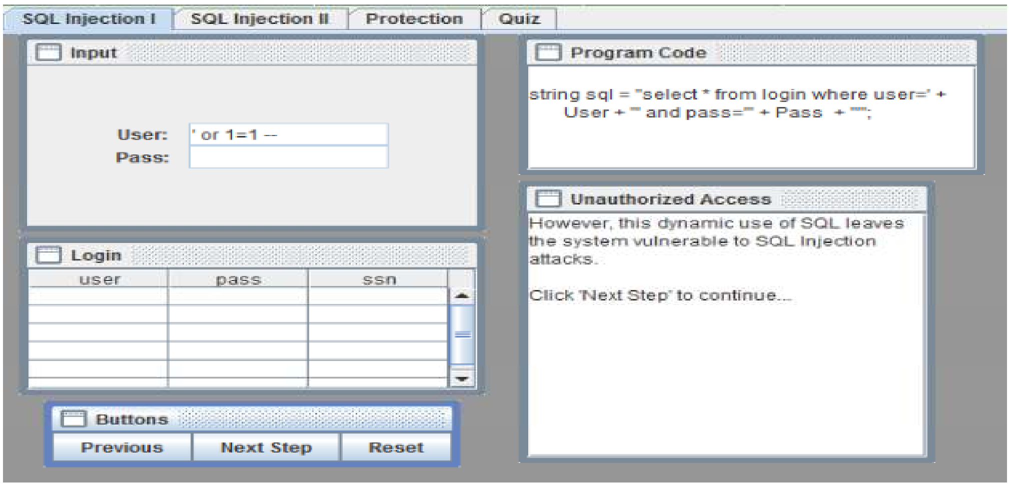
AND Password=‘‘;

The expression 1 = 1 is true for every row in the table causing the OR clause to return a value of true. The double hyphens comment out the rest of the SQL query string. This query will return a count greater than zero, assuming there is at least one row in the users table, resulting in what appears to be a successful login. In fact, it is not. Access to the system was successful without a user having to know either a username or password. Another SQL injection is made possible when a database system allows for the processing of stacked queries. Stacked queries are the execution of more than one SQL query in a single function call from an application program. In his case, one string is passed to the database system with multiple queries, each separated by a semicolon. The following example demonstrates a stacked query. The original intent is to allow the user to select attributes of products retained in a Products table. The user injects a stacked query incorporating an additional SQL query that also deletes the Customers table.

SELECT * FROM PRODUCTS; DROP CUSTOMERS;

This string when passed as an SQL query will result in the execution of two queries. A listing of all information for all products will be returned. In addition the Customers table will be removed from the database. The table structure will be deleted and all customer data will be lost. In database systems that do not allow stacked queries, or invalidate SQL strings containing a semicolon this query would not be executed.

The ADbC courseware sub-module for SQL injections demonstrates the insertion of malicious code during the login process. The sub-module steps through the process by first showing the entry of valid data and then demonstrating entry of malicious code, how it is injected into a dynamically created SQL statement and then executed. Figure 5 shows the step where malicious code is entered. Figure 6 shows the dynamically created SQL command and the resulting display of all the data in the user table. Additional steps present code resulting in the modification or deletion of data.



User: ' or 1=1 --
Pass:

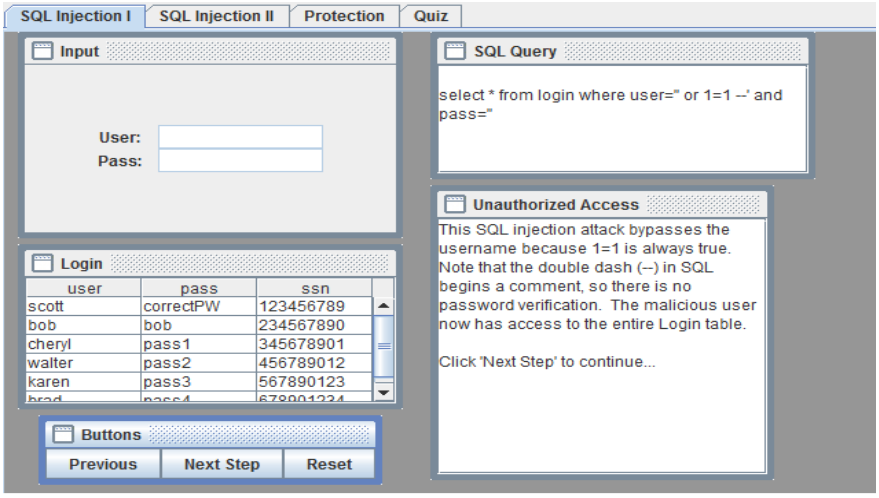
```
string sql = "select * from login where user=' +
      User + '" and pass='" + Pass  + "'";
```

user	pass	ssn

However, this dynamic use of SQL leaves the system vulnerable to SQL Injection attacks.

Click 'Next Step' to continue...

Figure 6: ADbC SQL Injection Sub-Module: Result of SQL Injection using Malicious Code



```
select * from login where user='' or 1=1 --' and
pass=''
```

user	pass	ssn
scott	correctPW	123456789
bob	bob	234567890
cheryl	pass1	345678901
walter	pass2	456789012
karen	pass3	567890123
brad	pass4	678901234

This SQL injection attack bypasses the username because 1=1 is always true. Note that the double dash (--) in SQL begins a comment, so there is no password verification. The malicious user now has access to the entire Login table.

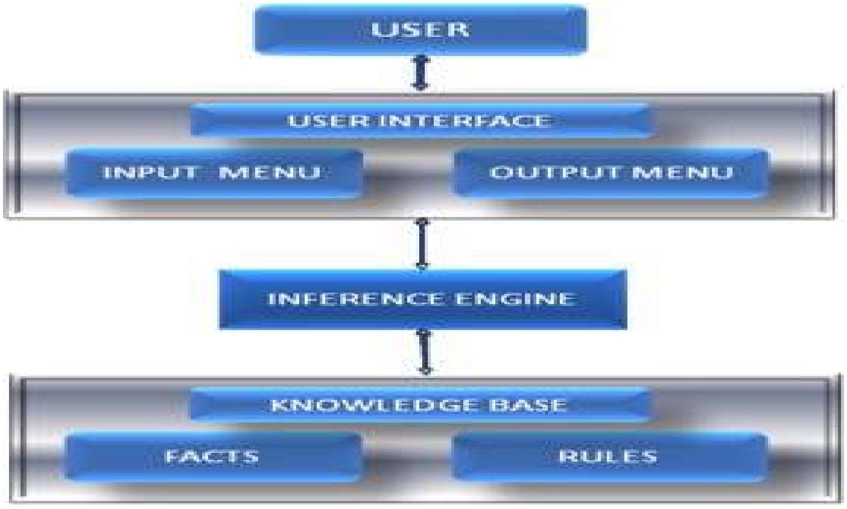
Click 'Next Step' to continue...

Figure 6: ADbC SQL Injection Sub-Module: Result of SQL Injection using Malicious Code

SQL injection vulnerabilities result from the dynamic creation of SQL queries in application programs that access a database system. The SQL queries are built incorporating user input and passed to the database system as a string variable. SQL injections can be prevented by validating user input. Three approaches are commonly used to address query string validation: using a black list, using a white list, or implementing parameterized queries. The black list parses the input string comparing each character to a predefined list of non-allowed characters. The disadvantage to using a black list is that many special characters can be legitimate but will be rejected using this approach. The common example is the use of the apostrophe in a last name such as O'Hare. The white list approach is similar except that each character is compared to a list of allowable characters. The approach is preferred but special considerations have to be made when validating the single quote. Parameterized queries use internally defined parameters to fill in a previously prepared SQL statement. The importance of input validation cannot be overstated. It is one of the primary defense mechanisms for preventing database vulnerabilities including SQL injections.

DATABASE INFERENCE

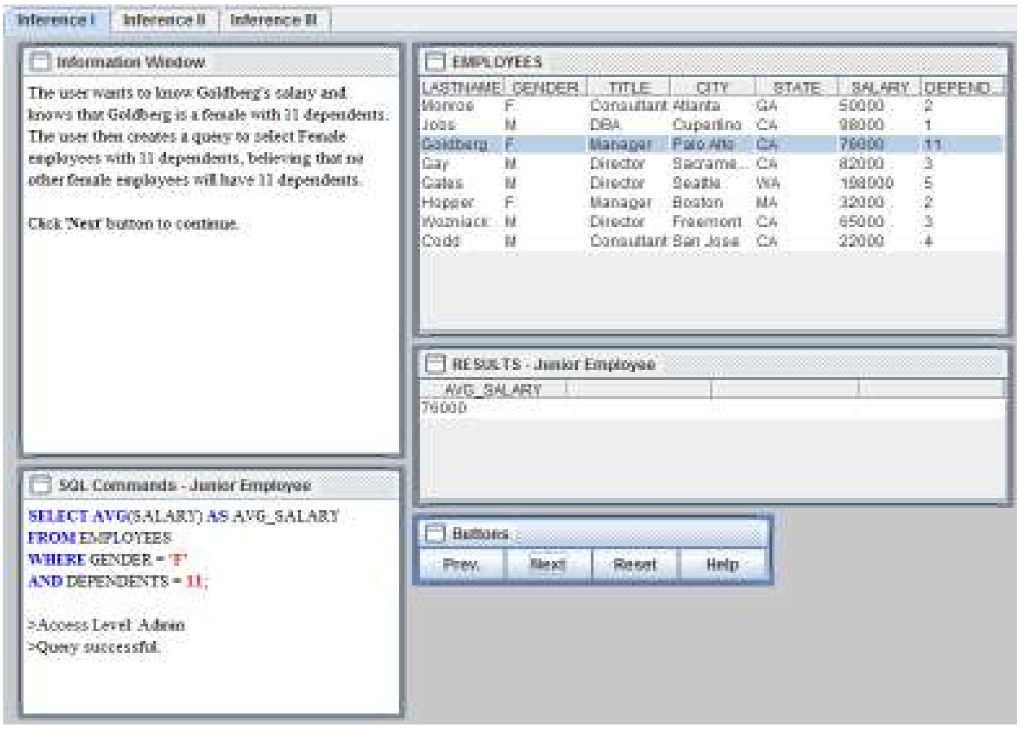
A subtle vulnerability found within database technologies is inference, or the ability to derive unknown information based on retrieved information. The problem with inference is that there are no ideal solutions to the problem. The only recommended solutions include controls related to queries (suppression) or controls related to individual items in a database (concealing). In other words, sensitive data requested in a query are either not provided or answers given are close but not exact, preventing the user from obtaining enough information to make inferences. Neither of these represents ideal solutions as they are restrictive in nature. However, it is important for students to understand the risks of inference and how it might occur. Examples are the best way to demonstrate inference.



The ADbC inference sub-module includes three animations that demonstrate how users might be able to put together (infer) information when data is available to those with a higher security access level or when they are only given access to aggregate data. Inference often happens in cases where the actual intent is for users to generate or view aggregate values when they have not been given access to individual data items. However, because they are exposed to information about the data, they are sometimes able to infer individual data values. Take for example a scenario where a worker desires to find out their co-worker Goldberg's salary In this organization, salary data is confidential. The worker has rights to generate aggregate data such as summarizing organizational salary data averaged across specific criteria (i.e., salary averaged by gender). Although the worker does not have access to individual data items, he or she does possess particular and unique details about Goldberg; specifically that Goldberg is a female and has 11 dependents. Based on this information, the worker can derive an aggregate function such as

SELECT AVG (Salary) FROM EMPLOYEES WHERE Gender = “F” and Dependents = 11.

This will return Goldberg’s salary because the average is taken from an aggregated data set of one. The ADbC inference sub-module animation for this scenario is illustrated in Figure 7. The SQL-command window depicts the construction of the requested query to ascertain salary averages. Employees Table data is shown in the upper left and underneath is the result of the query



The user wants to know Goldberg's salary and knows that Goldberg is a female with 11 dependents. The user then creates a query to select Female employees with 11 dependents, believing that no other female employees will have 11 dependents.

Click 'Next' button to continue.

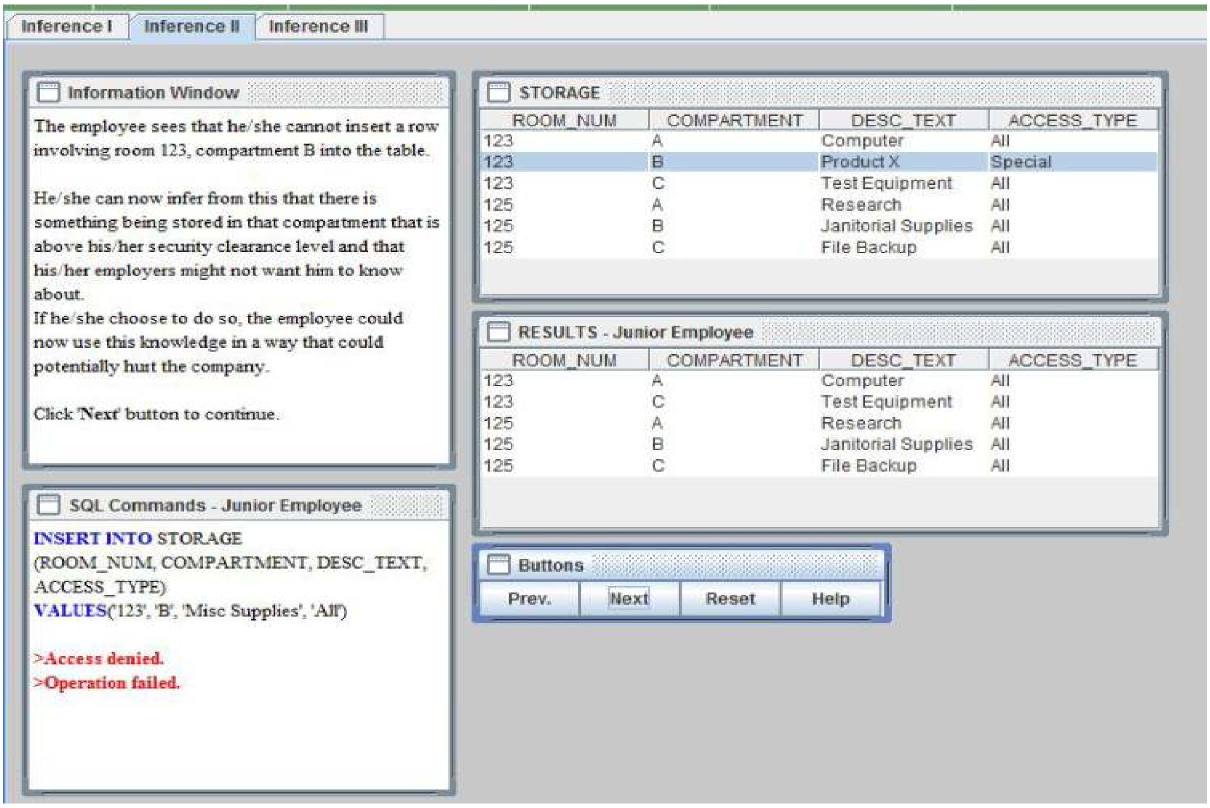
LASTNAME	GENDER	TITLE	CITY	STATE	SALARY	DEPEND...
Monroe	F	Consultant	Atlanta	GA	50000	2
Jobs	M	DBA	Cupertino	CA	98000	1
Goldberg	F	Manager	Palo Alto	CA	76000	11
Gay	M	Director	Sacrame...	CA	82000	3
Gates	M	Director	Seattle	WA	198000	5
Hopper	F	Manager	Boston	MA	32000	2
Wozniack	M	Director	Freemont	CA	65000	3
Codd	M	Consultant	San Jose	CA	22000	4

AVG_SALARY
76000

```
SELECT AVG(SALARY) AS AVG_SALARY
FROM EMPLOYEES
WHERE GENDER = 'F'
AND DEPENDENTS = 11;

>Access Level: Admin
>Query successful.
```

Figure 7. ADbC Inference Sub-module: Using Aggregate Data to Infer Information

Inference can also occur when users are able to ascertain information from data accessible to them at their security level even though that specific information is protected at a higher security access level. It is difficult to explain this without the aid of a demonstration. The second example in the ADbC Inference sub-module provides a scenario where specific data, in this case company product prototype data, is not made accessible to junior employees. However, junior employees are given access to update the Storage table that tracks the contents in company storage areas. When perusing this table, the junior employee is not able to read any rows containing prototype products. The problem occurs if the employee tries to update a protected row. This triggers an error message. Based on the error message, the junior employee could surmise that information was being hidden and might infer that something of a secretive nature was being stored in the storage compartment referenced in the update request. Figure 8 depicts an error being generated when a junior employee issues a query against a protected row of data. The table on the top right shows all of the data contained in the Storage table. The table on the bottom shows the data accessible to junior employees. Notice that Compartment B containing ProductX is not displayed in the lower table. A possible solution to this inference problem is polyinstantiation. Polyinstantiation allows a database to retain multiple records having the same primary key; they are uniquely distinguished by a security level identifier. If polyinstantiation were enacted in the proceeding scenario, the insert would be successful. However, this does not prevent the 'double booking' of the storage compartment area.



Information Window

The employee sees that he/she cannot insert a row involving room 123, compartment B into the table.

He/she can now infer from this that there is something being stored in that compartment that is above his/her security clearance level and that his/her employers might not want him to know about.
If he/she choose to do so, the employee could now use this knowledge in a way that could potentially hurt the company.

Click 'Next' button to continue.

STORAGE

ROOM_NUM	COMPARTMENT	DESC_TEXT	ACCESS_TYPE
123	A	Computer	All
123	B	Product X	Special
123	C	Test Equipment	All
125	A	Research	All
125	B	Janitorial Supplies	All
125	C	File Backup	All

RESULTS - Junior Employee

ROOM_NUM	COMPARTMENT	DESC_TEXT	ACCESS_TYPE
123	A	Computer	All
123	C	Test Equipment	All
125	A	Research	All
125	B	Janitorial Supplies	All
125	C	File Backup	All

SQL Commands - Junior Employee

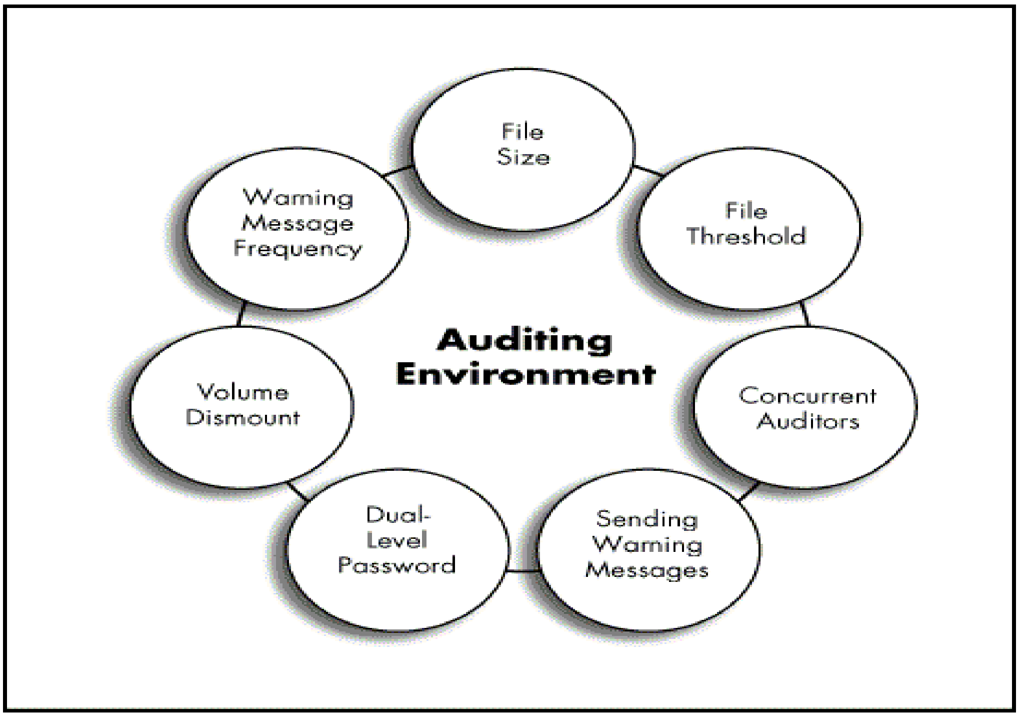
```
INSERT INTO STORAGE
(ROOM_NUM, COMPARTMENT, DESC_TEXT,
ACCESS_TYPE)
VALUES('123', 'B', 'Misc Supplies', 'All')

>Access denied.
>Operation failed.
```

Figure 8. ADbC Inference Sub-module: Security Level Access Error Leading to Inference

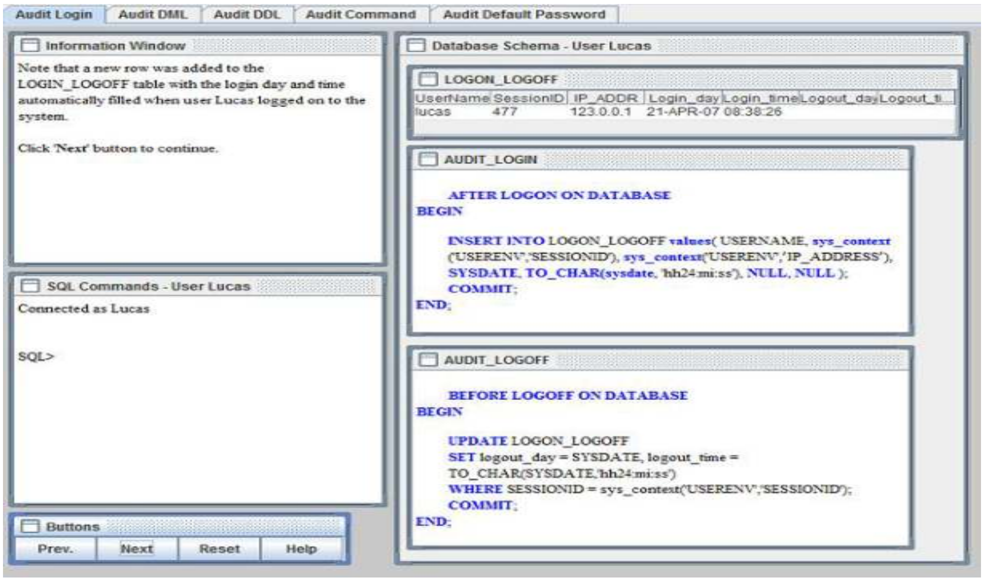
Developing technological solutions to detecting database inference is complex. Much of the work done in this area involves revoking access to specific database objects based on a user's past querying history (Staddon, 2003). The problem with inference detection, especially when done at query processing time, is that it results in a significant delay between the time the query is executed and the results are presented. As with other approaches to mitigating database security vulnerabilities, trade-offs must be made. The protection of highly sensitive data requires an examination of what situations could lead to exposure to unauthorized users and what monitoring policies should be implemented to insure appropriate responses are enacted.

AUDITING

Database auditing is used to track database access and user activity. Auditing can be used to identify who accessed database objects, what actions were performed, and what data was changed. Database auditing does not prevent security breaches, but it does provide a way to identify if breaches have occurred. Common categories of database auditing include monitoring database access attempts, Data Control Language (DCL) activities, Data Definition Language (DDL) activities, and Data Manipulation Language (DML) activities (Yang, 2009). Monitoring access attempts includes retaining information on successful and unsuccessful logon and logoff attempts. DCL audits record changes to user and role privileges, user additions, and user deletions. DDL audits record changes to the database schema such as changes to table structure or attribute datatypes. DML audits record changes to data. In addition, database errors should be monitored (Yang, 2009).



Database auditing is implemented via log files and audit tables. The real challenge of database auditing is deciding what and how much data to retain and how long to keep it. Several options exist. A basic audit trail usually captures user access, system resources used, and changes made to the structure of a database. More complete auditing captures data reads as well as data modifications. The ADbC auditing sub-module provides step-by-step examples for creating audits of user sessions, changes to database structure, and modifications to data. Figure 9 shows an example of code required to implement and trigger an audit of a user login. Data recorded includes the username and the date and time of the user login and



Information Window

Note that a new row was added to the LOGIN_LOGOFF table with the login day and time automatically filled when user Lucas logged on to the system.

Click 'Next' button to continue.

SQL Commands - User Lucas

```
Connected as Lucas


SQL>
```

Database Schema - User Lucas

LOGON_LOGOFF

UserName	SessionID	IP_ADDR	Login_day	Login_time	Logout_day	Logout_ti
lucas	477	123.0.0.1	21-APR-07	08:38:26		

AUDIT_LOGIN

```
    AFTER LOGON ON DATABASE
BEGIN

    INSERT INTO LOGON_LOGOFF values( USERNAME, sys_context
    ('USERENV','SESSIONID'), sys_context('USERENV','IP_ADDRESS'),
    SYSDATE, TO_CHAR(sysdate, 'hh24:mi:ss'), NULL, NULL );
    COMMIT;
END;
```

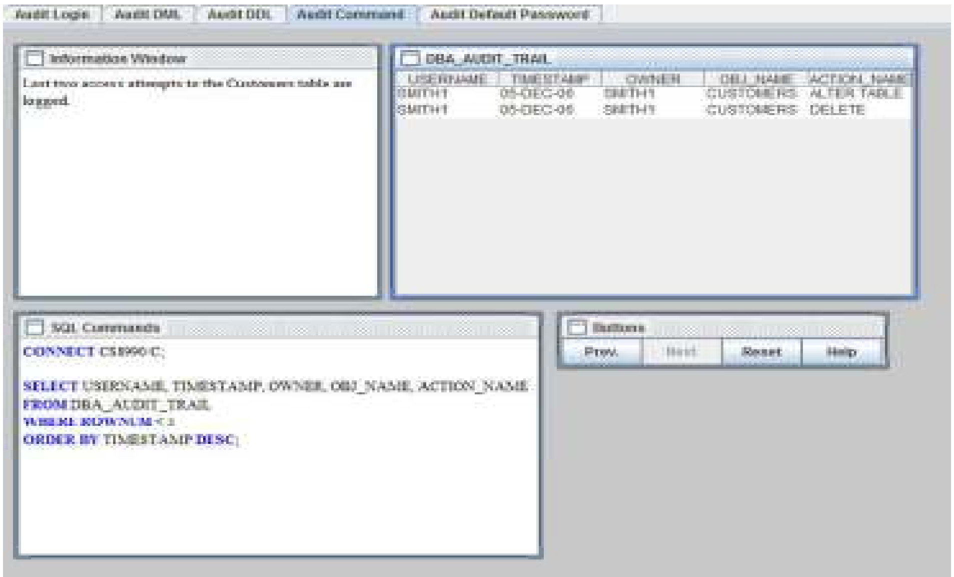
AUDIT_LOGOFF

```
    BEFORE LOGOFF ON DATABASE
BEGIN

    UPDATE LOGON_LOGOFF
    SET logout_day = SYSDATE, logout_time =
    TO_CHAR(SYSDATE,'hh24:mi:ss')
    WHERE SESSIONID = sys_context('USERENV','SESSIONID');
    COMMIT;
END;
```

Figure 9. ADbC Database Audit Sub-module: Monitoring User Logins

An audit trail provides a more complete trace recording of not only user access but also user actions.

This type of facility is included with many database management systems. The most common items that are audited include login attempts, data read and data modifications operations, unsuccessful attempts to access database tables, and attempts to insert data that violates specific constraints. Figure 10 shows an example audit trail of user access and user actions as demonstrated in the Audit Command animation in the ADbC Database Audit sub-module. The SQL Commands window displays the SQL statement used to retrieve data from the audit table.



Last two access attempts to the Customers table are logged.

USERNAME	TIMESTAMP	OWNER	OBJ_NAME	ACTION_NAME
SMITH1	05-DEC-06	SMITH1	CUSTOMERS	ALTER TABLE
SMITH1	05-DEC-06	SMITH1	CUSTOMERS	DELETE

```
CONNECT CS890/C;

SELECT USERNAME, TIMESTAMP, OWNER, OBJ_NAME, ACTION_NAME
FROM DBA_AUDIT_TRAIL
WHERE ROWNUM < 3
ORDER BY TIMESTAMP DESC;
```

Figure 10. ADbC Database Audit Sub-module: Example Database Audit Trail

Auditing plays a central role in a comprehensive database security plan. The primary weakness of the audit process is the time delay between when data is recorded and when analysis is performed. Consequently, breaches and other unauthorized activities are identified after the fact, making it difficult to mitigate adverse effects in a timely manner. However, solutions are being introduced that allow for real-time monitoring of database activity looking for patterned events indicative of potential breaches and enacting real-time notification to database administrators when such actions occur. Whatever the case, database auditing is a necessary process, and students must be made aware of the need for continuous monitoring of database log files.

CONCLUSION

The need to secure computer systems is well understood and securing data must be part of an overall computer security plan. Growing amounts of sensitive data are being retained in databases and more of these databases are being made accessible via the Internet. As more data is made available electronically, it can be assumed that threats and vulnerabilities to the integrity of that data will increase as well. Database security is becoming an increasingly important topic and students need to develop core understandings in this area. The primary objectives of database security are to prevent unauthorized access to data, prevent unauthorized tampering or modification of data, and to insure that data remains available when needed. The concepts related to database security are multifaceted. This makes it challenging to teach the material when database security is included as just one component of a larger course. However, this is how most students are exposed to the topic. This paper suggested a set of sub-topics in a database security course component and introduced a set of interactive software modules mapped to each sub-topic presented. Engaging students in interactive learning activities enhances the learning experience and provides the opportunity for students to further explore database security issues and identify practical implementation methods to database security mechanisms and strategies.

Secure Data Handling

Data handling related to when you view, update, delete, transfer, mail, store, or destroy data. It also relates to how you transfer the data from one location to another. Data is not always stored electronically. Occasionally it could be paper stored in a filing cabinet or in a binder.

Complying with secure best practices when identifying, transmitting, redistributing, storing or disposing of restricted data.

- **Authentication:** The process of identifying an individual, usually based on a username and password. In security systems, authentication is distinct from authorization , which is the process of giving individuals access to system objects based on their identity. Authentication merely ensures that the individual is who he or she claims to be, but says nothing about the access rights of the individual.
- **Encryption:** Encryption is the conversion of data into a form, called a ciphertext, that cannot be easily understood by unauthorized people. Decryption is the process of converting encrypted data back into its original form, so it can be understood.

The use of encryption/decryption is as old as the art of communication. In wartime, a cipher, often incorrectly called a code, can be employed to keep the enemy from obtaining the contents of transmissions. (Technically, a code is a means of representing a signal without the intent of keeping it secret; examples are Morse code and ASCII.) Simple ciphers include the substitution of letters for numbers, the rotation of letters in the alphabet, and the "scrambling" of voice signals by inverting the sideband frequencies. More complex ciphers work according to sophisticated computer algorithms that rearrange the data bits in digital signals.

In order to easily recover the contents of an encrypted signal, the correct decryption key is required. The key is an algorithm that undoes the work of the encryption algorithm. Alternatively, a computer can be used in an attempt to break the cipher. The more complex the encryption algorithm, the more difficult it becomes to eavesdrop on the communications without access to the key.

Encryption/decryption is especially important in wireless communications. This is because wireless circuits are easier to tap than their hard-wired counterparts. Nevertheless, encryption/decryption is a good idea when carrying out any kind of sensitive transaction, such as a credit-card purchase online, or the discussion of a company secret between different departments in the organization. The stronger the cipher -- that is, the harder it is for unauthorized people to break it -- the better, in general. However, as the strength of encryption/decryption increases, so does the cost.

In recent years, a controversy has arisen over so-called strong encryption. This refers to ciphers that are essentially unbreakable without the decryption keys. While most companies and their customers view it as a means of keeping secrets and minimizing fraud, some governments view strong encryption as a potential vehicle by which terrorists might evade authorities. These governments, including that of the United States, want to set up a key-escrow arrangement. This means everyone who uses a cipher would be required to provide the government with a copy of the key. Decryption keys would be stored in a supposedly secure place, used only by authorities, and used only if backed up by a court order. Opponents of this scheme argue that criminals could hack Into the key-escrow database and illegally obtain, steal, or alter the keys. Supporters claim that while this is a possibility, implementing the key escrow scheme would be better than doing nothing to prevent criminals from freely using encryption/decryption.

- **Transaction security:**
- **Audit trail:** A record showing who has accessed a computer system and what operations he or she has performed during a given period of time. Audit trails are useful both for maintaining security and for recovering lost transactions. Most accounting systems and database management systems include an audit trail component. In addition, there are separate audit trail software products that enable network administrators to monitor use of network resources.
- **Physical security:** Physical security describes security measures that are designed to deny access to unauthorized personnel (including attackers or even accidental intruders) from physically accessing a building, facility, resource, or stored information; and guidance on how to design structures to resist potentially hostile acts.[1] Physical security can be as simple as a locked door or as elaborate as multiple layers of barriers, armed security guards and guardhouse placement

Secure storage: Storage security is the group of parameters and settings that make storage resources available to authorized users and trusted networks - and unavailable to other entities. These parameters can apply to hardware, programming, communications protocols, and organizational policy.

Several issues are important when considering a security method for a storage area network (SAN). The network must be easily accessible to authorized people, corporations, and agencies. It must be difficult for a potential hacker to compromise the system. The network must be reliable and stable under a wide variety of environmental conditions and volumes of usage. Protection must be provided against online threats such as viruses, worms, Trojans, and other malicious code. Sensitive data should be encrypted. Unnecessary services should be disabled to minimize the number of potential security holes. Updates to the operating system, supplied by the platform vendor, should be installed on a regular basis. Redundancy, in the form of identical (or mirrored) storage media, can help prevent catastrophic data loss if there is an unexpected malfunction. All users should be informed of the principles and policies that have been put in place governing the use of the network.

Two criteria can help determine the effectiveness of a storage security methodology. First, the cost of implementing the system should be a small fraction of the value of the protected data. Second, it should cost a potential hacker more, in terms of money and/or time, to compromise the system than the protected data is worth.

Data backup: Database backup is the process of making a complete secondary copy of a database or database server for the purpose of recovering the database following a disaster. Businesses who rely upon databases to conduct business operations and/or provide services are especially susceptible to the loss of database information, and therefore generally invest in some form of database backup. Most database management systems (DBMSs) feature the ability to create a backup of a given instance locally; however, local data remains susceptible to loss or damage from accidental deletion, drive or partition errors, or physical damage due to hardware failure or disaster. A properly conducted database backup ensures that the most recent copy of operational data is available for recovery.

Backup service providers such as CRC offer the ability to make complete backups offsite of common database servers such as SQL Server, Oracle and DB2. In addition, CRC allows users to perform a database backup without taking the database server offline, a critical consideration when database functionality is critical to core business tasks.

Databases are also regularly backed up for reporting and compliance purposes, since regulations imposed by the Federal goverment require many types of organizations to maintain access to electronic records.

Database backup from providers like CRC also involve information lifecycle management (ILM), transferring inactive or less frequently-accessed data to tiered storage, reducing the amount of time involved in a backup while freeing up resources to improve the performance of database-driven applications.

Secure Connection

a) Wired and wireless:

Wired: why would you use it?

A wired home network is when you connect your computer or other compatible device to your Virgin Media Hub or Super Hub with an Ethernet cable. With the introduction of wireless routers, it's less common these days for a home to only use a wired network.

The best thing about a wired connection is the reliability and speed it gives you (wired is faster than wireless). This makes it ideal for things that use a lot of bandwidth, like playing online games on your Xbox.

What's great about wired?

- Faster and more reliable than wireless connections
- Less risk of others being able to access your broadband connection
- Easy to set up and troubleshoot
- No extra wireless equipment is needed

What's not?

- Not so flexible when positioning computers and devices because
- you need to be connected to your Hub using an Ethernet cable
- Not so convenient for users of laptops and other mobile devices
- Supports fewer connections than a wireless.

Wireless: why would you use it?

A wireless network is when your computer (or other wireless compatible device) is connected to your Virgin Media Hub or Super Hub without any wires.

Want to check your emails on your laptop in the garden? No problem. Want to tell your friends what you've been up to on Facebook? You can do that too.

And because you can connect lots of different devices to the web, everyone in your home can go online at the same time.

What's great about wireless?

- No wires! So you can connect wherever you want
- Ideal for users of laptops and other mobile devices
- You can connect more devices than via a wired connection
- Very secure when used with the highest strength security settings (WPA encryption - which comes as standard on our Super Hub)
- You can connect your smartphone to your wireless network for faster browsing

What's not?

- Performance can be affected by thick walls, electrical interference etc
- It can be up to 30% slower than wired connections
- Un authorised users might try to use your connection (which is why security is so important)
- If you have an older computer, a wireless USB adapter may be needed

b) **Protocols:** An agreed-upon format for transmitting data between two devices. The protocol determines the following:
   - the type of error checking to be used
   - data compression method, if any

how the sending device will indicate that it has finished sending a message how the receiving device will indicate that it has received a message

There are a variety of standard protocols from which programmers can choose. Each has particular advantages and disadvantages; for example, some are simpler than others, some are more reliable, and some are faster.

From a user's point of view, the only interesting aspect about protocols is that your computer or device must support the right ones if you want to communicate with other computers. The protocol can be implemented either in hardware or in software.

c) **Monitor for attacks:** Local cellular operators monitor traffic in a bid to pick up and stop distributed denial of service (DDOS) attacks, before they have a major impact on service levels to subscribers.

   DDOS attacks make use of multiple compromised systems, often infected with a Trojan, to target a single system, which leads to a denial of service attack. Victims of these attacks include the end targeted system and all systems maliciously used and controlled by the hacker.

   The only way to pick up the intrusion is to monitor the traffic in real-time to detect unusual patterns, says Fryer. He adds that most mobile operators and Internet service providers have deployed a DDOS mechanism, which forms part of a global DDOS monitoring centre.

   Most of the deployments have updated blacklisting capabilities to fend off attackers, says Fryer. He adds that real-time monitoring is more effective, and allows Vodacom to block the source IP, although hackers then come back with a different address.

d) **Virus:** A computer virus is a computer program that can replicate itself and spread from one computer to another. The term "virus" is also commonly, but erroneously, used to refer to other types of malware, including but not limited to adware and spyware programs that do not have a reproductive ability.

Malware includes computer viruses, computer worms, ransomware, trojan horses, keyloggers, most rootkits, spyware, dishonest adware, malicious BHOs and other malicious software. The majority of active malware threats are usually trojans or worms rather than viruses. Malware such as trojan horses and worms is sometimes confused with viruses, which are technically different: a worm can exploit security vulnerabilities to spread itself automatically to other computers through networks, while a trojan horse is a program that appears harmless but hides malicious functions. Worms and trojan horses, like viruses, may harm a computer system's data or performance. Some viruses and other malware have symptoms noticeable to the computer user, but many are surreptitious or simply do nothing to call attention to themselves. Some viruses do nothing beyond reproducing themselves.

Denial of service : In computing, a denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack) is an attempt to make a machine or network resource unavailable to its intended users. Although the means to carry out, motives for, and targets of a DoS attack may vary, it generally consists of efforts to temporarily or indefinitely interrupt or suspend services of a host connected to the Internet.

Perpetrators of DoS attacks typically target sites or services hosted on high-profile web servers such as banks, credit card payment gateways, and even root nameservers. This technique has now seen extensive use in certain games, used by server owners, or disgruntled competitors on games such as Minecraft. The term is generally used relating to computer networks, but is not limited to this field; for example, it is also used in reference to CPU resource management.

One common method of attack involves saturating the target machine with external communications requests, so much so that it cannot respond to legitimate traffic, or responds so slowly as to be rendered essentially unavailable. Such attacks usually lead to a server overload. In general terms, DoS attacks are implemented by either forcing the targeted computer(s) to reset, or consuming its resources so that it can no longer provide its intended service or obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

Denial-of-service attacks are considered violations of the Internet Architecture Board's Internet proper use policy, and also violate the acceptable use policies of virtually all Internet service providers. They also commonly constitute violations of the laws of individual nations.

Man in the middle: The man-in-the-middle attack (often abbreviated MITM, MitM, MIM, MiM, MITMA, also known as a bucket brigade attack, or sometimes Janus attack) in cryptography and computer security is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all messages going between the two victims and inject new ones, which is straightforward in many circumstances (for example, an attacker within reception range of an unencrypted Wi-Fi wireless access point, can insert himself as a man-in-the-middle).

This maneuver precedes computers. A fictional example of a "man-in-the-middle attack" utilizing a telegraph is featured in the 1898 short story The Man Who Ran Europe by Frank L. Pollack.

A man-in-the-middle attack can succeed only when the attacker can impersonate each endpoint to the satisfaction of the other — it is an attack on mutual authentication (or lack thereof). Most cryptographic protocols include some form of endpoint authentication ecifically to prevent MITM attacks. For example, SSL can authenticate one or both parties using a mutually trusted certification authority.

f) **Trusted IP addresses:** When you set up your secure zone you can add trusted IP addresses to is by selecting 'Trusted IP Addresses' menu item. When you trust one or more IP addresses, any visitor accessing secure content inside your secure zone will gain instant access to that content without needing to be registered or log in. Its a great feature for intranet setups such as internal faculty and staff pages in universities, or even corporate directories.

g) **Role-based security:** Role-based security is a principle by which developers create systems that limit access or restrict operations according to a user's constructed role within a system. This is also often called role-based access control, since many businesses and organizations use this principle to ensure that unauthorized users do not gain access to privileged information within an IT architecture.

There are many ways to develop a role-based security system. All of them start with the definition of various roles and what users assigned to those roles can and can't do or see. The resulting levels of functionality must be coded into the system using specific parameters.

Object-oriented programming often involves treating a role as an object relative to certain code modules or functions. In a Microsoft programming setting, a developer might use a PrincipalPermission object in .Net to examine an object containing a role designation and to perform security checks. In other cases, information about an object can be passed to a method for a security check.

Any role-based security system depends on the code's ability to correctly and thoroughly control a given user by his or her assigned role and therefore guard against unauthorized use of proprietary identifiers of a specific role. Alternative models include mandatory access control, where certain specifics are coded into an operating system, and discretionary access control, where some elements of security may be more flexible. For example, a more privileged user may be able to "pass" access to another user in a simple discretionary event or process.

h) **Port monitorng:** Port mirroring is a method of copying and sending network packets transmitted as input from a port to another port of a monitoring computer/switch/device. It is a network monitoring technique implemented on network switches and similar devices.

Port mirroring is implemented in local area networks (LAN), wireless local area networks (WLAN) or virtual local area networks (VLAN) to identify, monitor and troubleshoot network abnormalities. It is configured at the network switch by a network administrator (NA) or network monitoring/security application. When enabled, the traffic that emerges to and from a specific port number is automatically copied and transmitted to a monitoring/destination port. Typically, the destination port is part of the monitoring software or security application that analyzes these data packets.

The port mirroring process is generally hidden from the source and other nodes on the network.

Port mirroring is also known as switched port analyzer (SPAN) and roving analysis port (RAP).

Security Policies: Security policy is a definition of what it means to be secure for a system, organization or other entity. For an organization, it addresses the constraints on behavior of its members as well as constraints imposed on adversaries by mechanisms such as doors, locks, keys and walls. For systems, the security policy addresses constraints on functions and flow among them, constraints on access by external systems and adversaries including programs and access to data by people.

- **Organization-wide implementation:** There is a great need for organizational leaders to provide data-based evidence that a program or initiative makes a difference. The authors describe findings from a survey designed to gather baseline data about changes organizations experience after implementing the Clinical Practice Model framework, and report how the Clinical Practice Model Resource Center staff used the survey findings to build the capacity of individuals accountable for implementing this integrated, interdisciplinary professional practice framework into the organization's operations.
- **People buy-in for enforcement:**
- **Periodic review of policies in place:** This work provides a comprehensive analysis of a general periodic review production/inventory model with random (variable) yield. Existence of an order point whose value does not depend on yield being random is proved in the single period case without specifying the yield model and using a very general cost structure. When yield is a random multiple of lot size, the nonorder-up-to optimal policy is characterized for a finite-horizon model. The finite-horizon value functions are shown to converge to the solution of an infinite-horizon functional equation, and the infinite-horizon order point is shown to be no smaller than when yield is certain.

UNIT-2

SECURITY MODEL-1

2.1 ACCESS MATRIX MODEL:

The access matrix model is the policy for user authentication, and has several implementations such as access control lists (ACLs) and capabilities. It is used to describe which users have access to what objects.

The access matrix model consists of four major parts:

A list of objects

A list of subjects

A function T which returns an object's type

The matrix itself, with the objects making the columns and the subjects making the rows In the cells where a subject and object meet lie the rights the subject has on that object. Some example access rights are read, write, execute, list and delete.

Example Access Matrix:

	Objects	
Subjects	**index.htmlfile**	**Java VM** **Virtual Machine**
John Doe	rwld	x
Sally Doe	rl	-

An access matrix has several standard operations associated with it:

i. Entry of a right into a specified cell

ii. Removal of a right from a specified cell

iii. Creation of a subject

iv. Creation of an object

v. Removal of an subject

vi. Removal of an object

Implementation:

The two most used implementations are access control lists and capabilities. Access control lists are achieved by placing on each object a list of users and their associated rights to that object. An interactive demonstration of access control lists can be seen here. For example, if we have file1, file2 and file3, and users (subjects) John and Sally, an access control list might look like:

	Objects (Files)		
Users	File1	File2	File3
John	RWX	R-X	RW-
Sally	---	RWX	R--

The rights are R (Read), W (Write) and X (Execute). A dash indicates the user does not have that particular right. Thus, John does not have permission to execute File3, and Sally has no rights at all on File1.

Users			
John	file1:RWX	file2:R-X	file3: RW-
Sally	file1: ---	file1:RWX	file1: R--

Capabilities are accomplished by storing on each subject a list of rights the subject has for every object. This effectively gives each user a keyring. To remove access to a particular object, every user (subject) that has access to it must be "touched". A touch is an examanition of a user's rights to that object and potentially removal of rights.

This brings back the problem of sweeping changes in access rights. Here is what an implementation of capabilities might look like, using the above example:

Access restrictions such as access control lists and capabilities sometimes are not enough. In some cases, information needs to be tightened further, sometimes by an authority higher than the owner of the information. For example, the owner of a top secret document in a government office might deem the information available to many users, but his manager might know the information should be restricted further than that. In this case, the flow of information needs to be controlled -- secure information cannot flow to a less secure user.

2.2 TAKE-GRANT MODEL:

The take-grant protection model is a formal model used in the field of computer security to establish or disprove the safety of a given computer system that follows specific rules. It shows that for specific systems the question of safety is decidable in linear time, which is in general un decidable.

The model represents a system as directed graph, where vertices are either subjects or objects. The edges between them are labelled and the label indicates the rights that the source of the edge has over the destination. Two rights occur in every instance of the model: take and grant. They play a special role in the graph rewriting rules describing admissible changes of the graph.

There are a total of four such rules:

i. Take rule allows a subject to take rights of another object (add an edge originating at the subject)
ii. Grant rule allows a subject to grant own rights to another object (add an edge terminating at the subject)
iii. Create rule allows a subject to create new objects (add a vertex and an edge from the subject to the new vertex)
iv. Remove rule allows a subject to remove rights it has over on another object (remove an edge originating at the subject)

Preconditions for take(o,p,r): subject s has the right Take for o. object o has the right r on p.

Preconditions for grant(o,p,r): subject s has the right Grant for o. s has the right r on p.

Using the rules of the take-grant protection model, one can reproduce in which states a system can change, with respect to the distribution of rights. Therefore one can show if rights can leak with respect to a given safety model.

The Take-Grant protection model is a formal access control model, which represents transformation of rights and information between entities inside a protection system. This model was presented first by Jones et al. [8] to solve the “Safety Problem”. They showed that using Take-Grant model, the safety problem is decidable and also can be solved in linear time according to the number of subjects and objects of the system.

In this model the protection state is represented as a directed finite graph. In the graph, vertices are entities of the system and edges are labeled. Each label indicates the rights that the source vertex of the corresponding edge has over the destination vertex. Entities could be subjects (represented by ●), objects (represented by) or play the both roles (represented by ⊗). The set of basic access rights is denoted as R={t,g,r,w} which t, g, r and w respectively stand for take, grant, read, and write ac- cess rights. To model the rights transfer, Take-Grant protection model uses a set of rules called de-jure rules. These rules transfer the Take-Grant graph to a new state which reflects the modification of protection state in an actual system. The de-jure Network Vulnerability Analysis Through Vulnerability Take-Grant Model (VTG) rules are take, grant, create and remove. The take and grant rules are described briefly as:

1. Take rule: Let x, y, and z be three distinct vertices in a protection graph G0 and let x be a subject. Let there is an edge from x to y labeled γ where t∈ γ, an edge from y to z labeled β. Then the take rule defines a new graph G1 by adding an edge to the protection graph from x to z labeled α, where α⊆β. Fig 1.(a) shows the take rule graphically.

2. Grant rule: Let x, y, and z be three distinct vertices in a protection graph G0 and let x be a subject. Let there is an edge from x to y labeled β where g∈ γ, an edge from x to z labeled β. Then the grant rule defines a new graph G1 by adding an edge to the protection graph from y to z labeled α, where α⊆β. Fig.1(b) shows the grant rule graphically.

Having the take right over another subject or object means that its owner can achieve all rights of the associated subject or object unconditionally. However, obtaining the rights through the grant rule requires cooperation of the grantor.

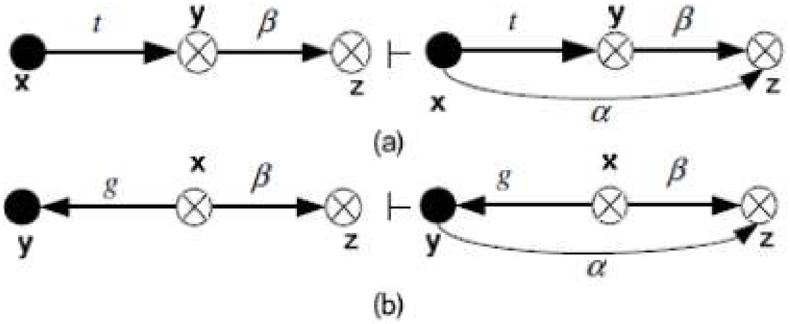


Fig. 1. (a) take rewriting rule. (b) grant rewriting rule.

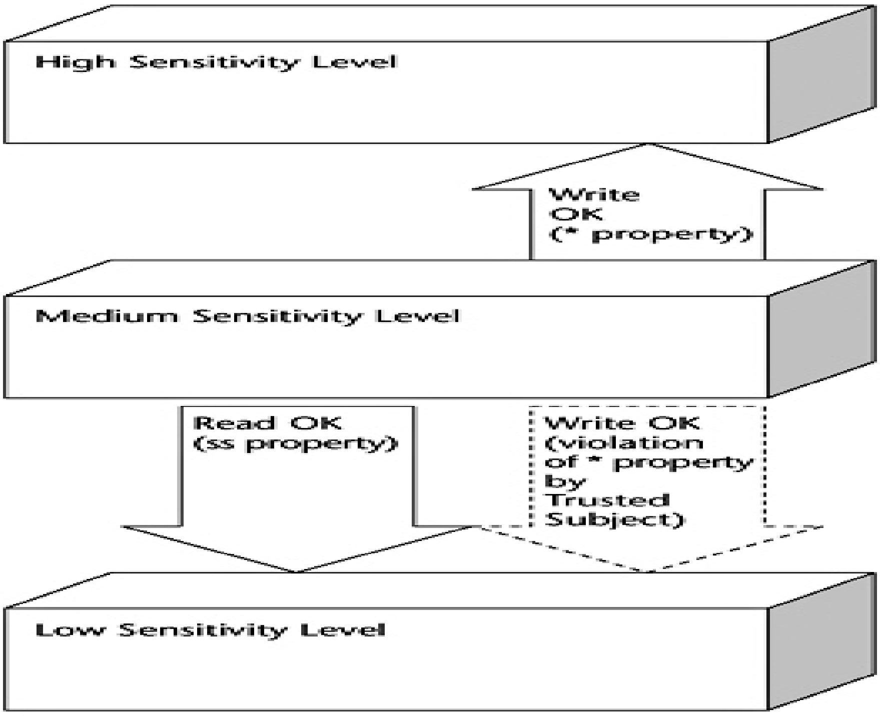
UNIT-3

SECURITY MODEL-2

3.1 BELL–LAPADULA MODEL:

The Bell–LaPadula Model (abbreviated BLP) is a state machine model used for enforcing access control in government and military applications. It was developed by David Elliott Bell and Leonard J. LaPadula, subsequent to strong guidance from Roger R. Schell to formalize the U.S. Department of Defence (DoD) multilevel security (MLS) policy. The model is a formal state transition model of computer security policy that describes a set of access control rules which use security labels on objects and clearances for subjects. Security labels range from the most sensitive (e.g."Top Secret"), down to the least sensitive (e.g., "Unclassified" or "Public").

The Bell–LaPadula model is an example of a model where there is no clear distinction of protection and security



FEATURES: The Bell–LaPadula model focuses on data confidentiality and controlled access to classified information, in contrast to the Biba Integrity Model which describes rules for the protection of data integrity. In this formal model, the entities in an information system are divided into subjects and objects. The notion of a "secure state" is defined, and it is proven that each state transition preserves security by moving from secure state to secure state, thereby inductively proving that the system satisfies the security objectives of the model. The Bell–LaPadula model is built on the concept of a state machine with a set of allowable states in a computer network system. The transition from one state to another state is defined by transition functions.

A system state is defined to be "secure" if the only permitted access modes of subjects to objects are in accordance with a security policy. To determine whether a specific access mode is allowed, the clearance of a subject is compared to the classification of the object (more precisely, to the combination of classification and set of compartments, making up the security level) to determine if the subject is authorized for the specific access mode. The clearance/classification scheme is expressed in terms of a lattice. The model defines two mandatory access control (MAC) rules and one discretionary access control (DAC) rule with three security properties:

i. The Simple Security Property - a subject at a given security level may not read an object at a higher security level (no read-up).

ii. The ★-property (read "star"-property) - a subject at a given security level must not write to any object at a lower security level (no write-down).

iii. The Discretionary Security Property - use of an access matrix to specify the discretionary access control.

The transfer of information from a high-sensitivity document to a lower-sensitivity document may happen in the Bell–LaPadula model via the concept of trusted subjects. Trusted Subjects are not restricted by the ★-property. Untrusted subjects are. Trusted Subjects must be shown to be trustworthy with regard to the security policy. This security model is directed toward access control and is characterized by the phrase: "no read up, no write down." Compare the Biba model, the Clark-Wilson model and the Chinese Wall model.

With Bell-LaPadula, users can create content only at or above their own security level (i.e. secret researchers can create secret or top-secret files but may not create public files; no write-down). Conversely, users can view content only at or below their own security level (i.e. secret researchers can view public or secret files, but may not view top-secret files; no read-up).

The Bell–LaPadula model explicitly defined its scope. It did not treat the following extensively:

i. Covert channels. Passing information via pre-arranged actions was described briefly.
ii. Networks of systems. Later modeling work did address this topic.
iii. Policies outside multilevel security. Work in the early 1990s showed that MLS is one version of boolean policies, as are all other published policies.

Strong ★ Property

The Strong ★ Property is an alternative to the ★-Property, in which subjects may write to objects with only a matching security level. Thus, the write-up operation permitted in the usual ★-Property is not present, only a write-to-same operation. The Strong ★ Property is usually discussed in the context of multilevel database management systems and is motivated by integrity concerns.[6] This Strong ★ Property was anticipated in the Biba model where it was shown that strong integrity in combination with the Bell–LaPadula model resulted in reading and writing at a single level.

TRANQUILITY PRINCIPLE:

The tranquility principle of the Bell–LaPadula model states that the classification of a subject or object does not change while it is being referenced. There are two forms to the tranquility principle: the "principle of strong tranquility" states that security levels do not change during the normal operation of the system. The "principle of weak tranquility" states that security levels may never change in such a way as to violate a defined security policy. Weak tranquility is desirable as it allows systems to observe the principle of least privilege. That is, processes start with a low clearance level regardless of their owners clearance, and progressively accumulate higher clearance levels as actions require it.

LIMITATIONS:

i. Only addresses confidentiality, control of writing (one form of integrity), ★-property and discretionary access control
ii. Covert channels are mentioned but are not addressed comprehensively
iii. The tranquility principle limits its applicability to systems where security levels do not change dynamically. It allows controlled copying from high to low via trusted subjects.

2.2 THE BIBA MODEL:

The Biba Model or Biba Integrity Model developed by Kenneth J. Biba in 1977,[1] is a formal state transition system of computer security policy that describes a set of access control rules designed to ensure data integrity. Data and subjects are grouped into ordered levels of integrity. The model is designed so that subjects may not corrupt objects in a level ranked higher than the subject, or be corrupted by objects from a lower level than the subject.

In general the model was developed to circumvent a weakness in the Bell–LaPadula model which only addresses data confidentiality.

FEATURES:

` In general, preservation of data integrity has three goals:

i. Prevent data modification by unauthorized parties
ii. Prevent unauthorized data modification by authorized parties
iii. Maintain internal and external consistency (i.e. data reflects the real world)

This security model is directed toward data integrity (rather than confidentiality) and is characterized by the phrase: "no read down, no write up". This is in contrast to the Bell-LaPadula model which is characterized by the phrase "no write down, no read up".

In the Biba model, users can only create content at or below their own integrity level (a monk may write a prayer book that can be read by commoners, but not one to be read by a high priest). Conversely, users can only view content at or above their own integrity level (a monk may read a book written by the high priest, but may not read a pamphlet written by a lowly commoner). Another analogy to consider is that of the military chain of command. A General may write orders to a Colonel, who can issue these orders to a Major. In this fashion, the General's original orders are kept intact and the mission of the military is protected (thus, "no read down" integrity). Conversely, a Private can never issue orders to his

Sergeant, who may never issue orders to a Lieutenant, also protecting the integrity of the mission ("no write up").

The Biba model defines a set of security rules similar to the Bell-LaPadula model. These rules are the reverse of the Bell-LaPadula rules:

i. The Simple Integrity Axiom states that a subject at a given level of integrity must not read an object at a lower integrity level (no read down).

ii. The * (star) Integrity Axiom states that a subject at a given level of integrity must not write to any object at a higher level of integrity (no write up).

SEA VIEW MODEL

The Sea View formal security policy model admits a range of designs for a multilevel secure relational database system. The requirement for a near-term implementation suggests that the design should utilize existing technology to the extent possible. Thus the design uses an existing database management system ported to an existing TCB (trusted computing base) environment. A pre processor translates key constructs of the Sea View multilevel relational data model to those of the standard relational model used by the commercial database system. The underlying reference monitor enforces mandatory and basic discretionary controls with A1 assurance. By combining single-level data into a multilevel view, it is possible to use a commercial database system and classify data at the relation level to implement the Sea View model, with element-level classification.

In Sea View the design approach is built on the notion of a reference monitor for mandatory security. Sea View provides the user with the basic abstraction of a multilevel relation in which the individual data elements are individually classified. This design approach implements multilevel relations as views stored over single level relations, transparent to the user. The single-level relations are stored in segments managed by an underlying mandatory reference monitor. This underlying mandatory reference monitor performs a label comparison for subjects and the segments for which they request access, to decide whether to grant access. The access class of any particular data element in a multilevel relation is derived from the access class of the single-level relation in which the data element is stored, which in turn matches the access class of the segment in which it is stored, which is known to the reference monitor. Thus, labels for each individual data element do not have to be stored, as was supposed prior to Sea View

In Sea View, every database function is carried out by a single-level subject. Thus, a database system subject, when operating on behalf of a user, cannot gain access to any data whose classification is not dominated by the user's clearance. The use of only single-level subjects for routine database operations provides the greatest degree of security possible and considerably reduces the risk of disclosure of sensitive data. This approach means that there must be at least one database server instance for each active access class. Thus, the database system consists of multiple database server instances that share the same logical database.

UNIT -4

SECURITY MECHANISM

4.1 INTRODUCTION IDENTIFICATION AND AUTHENTICATION OF USERS

Identification is the process whereby a network element recognizes a valid user's identity. Authentication is the process of verifying the claimed identity of a user. A user may be a person, a process, or a system (e.g., an operations system or another network element) that accesses a network element to perform tasks or process a call. A user identification code is a non-confidential auditable representation of a user. Information used to verify the claimed identity of a user can be based on a password, Personal Identification Number (PIN), smart card, biometrics, token, exchange of keys, etc. Authentication information should be kept confidential.

If users are not properly identified then the network element is potentially vulnerable to access by unauthorized users. Because of the open nature of ONA, ONA greatly increases the potential for unauthorized access. If strong identification and authorization mechanisms are used, then the risk that unauthorized users will gain access to a system is significantly decreased.

Section describes the threat of impersonating a user in more detail.

The exploitation of the following vulnerabilities, as well as other identification and authentication vulnerabilities, will result in the threat of impersonating a user.

Weak authentication methods are used;

The potential exists for users to bypass the authentication mechanism;

The confidentiality and integrity of stored authentication information is not preserved, and Authentication information which is transmitted over the network is not encrypted.

Computer intruders have been known to compromise PSN assets by gaining unauthorized access to network elements. It is possible for a person impersonating an authorized user to cause the full range of threats described in section . Impacts on the PSN caused by the threat of impersonating a user include the full range of impacts to NS/EP telecommunications described in section . The severity of the threat of impersonating a user depends on the level of privilege that is granted to the unauthorized user.

4.2 MEMORY PROTECTION:

Many embedded systems operate with a multitasking operating system which provides a facility to ensure that the task currently executing does not disrupt the operation of other tasks. System resources and the code and data of other tasks are protected. The protection system typically relies on both hardware and software to do this. In a system with no hardware protection support, each task must work in a cooperative way with other tasks and follow rules.

In contrast, a system with dedicated protection hardware will check and restrict access to system resources, preventing hostile or unintentional access to forbidden resources. Tasks are still required to follow a set of OS rules, but these are also enforced by hardware, which gives more robust protection.

ARM provides many processors with this capability, using either a memory protection unit (MPU) or a memory management unit (MMU). This Applications Note is about MPU based processors. These provide hardware protection over a number of software-programmed regions, but stop short of providing a full virtual memory system with address translation, which requires an MMU.

An ARM MPU uses regions to manage system protection. A region is a set of attributes associated with an area of memory. The processor core holds these attributes in CP15 registers and identifies each region with a number. A region’s memory boundaries are defined by its base address and its size. Each region possesses additional attributes which define access rights, memory type and the cache policies. Because peripherals are memory-mapped in ARM systems, the same protection mechanism is used for both system peripherals and task memory.

In the Cortex-R4 and Cortex-R5 processors, the presence of the MPU is optional although generally included. If present, there may be either 8, 12 or 16 such regions (defined by the hardware implementer at RTL configuration stage). The smallest length (size) of a region is just 32 bytes. If a region is of 256 bytes or more, it may be divided into 8 sub-regions. Although the Cortex-R4 and Cortex-R5 processors have a Harvard view of memory, the regions are common to both instruction and data accesses. However, it is possible to use the "Execute Never (XN)" attribute to disallow instructions execution from a peripheral or data region.

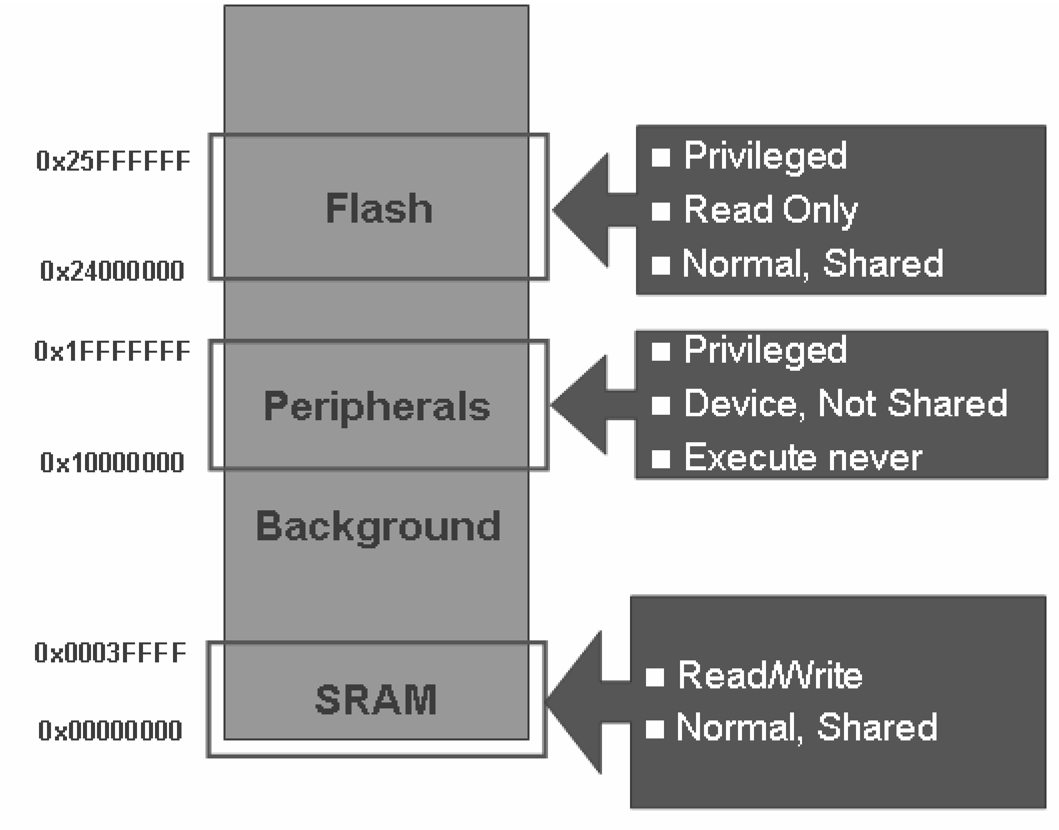


Figure MPU Region configuration with background region

Cache Maintenance Recommendations:

As we have seen, it will sometimes be necessary to perform cache maintenance operations (clean and/or invalidate) when changes are made to MPU region attributes.

Specifically, for the Cortex-R4 and R5 processors, it is changing from a less restrictive to a more restrictive attribute that will require cache maintenance, e.g. when changing a region's resolved attribute from cacheable to non-cacheable. Therefore, in the case of Cortex-R4 and R5 it is possible to identify changes to attributes 1 through 3 which would not necessitate cache maintenance.

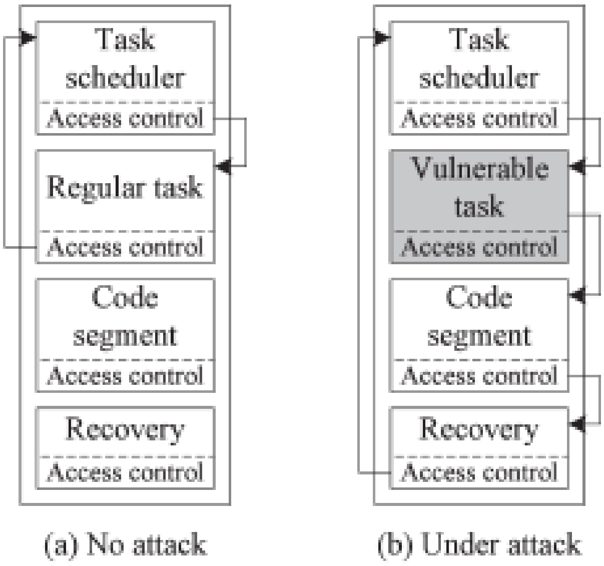
However, it is strongly recommended that cache is always maintained when changes are made to attributes 1 through 3 in order that programs remain platform independent. Additional implications may also exist in the level-2 memory system which are outside the domain of the processor and are system-specific.

Overall it is strongly recommended practice for any given memory location to have fixed values for attributes 1 through 3, and that these be independent of the currently executing context.

Failure to guarantee this will mean that the OS must explicitly manage the mismatched attributes, which will involve cache maintenance and other considerations. It should be emphasized that in most systems, attribute changes which require cache maintenance (changes to memory type or cache ability) do not typically occur after system start-up.

4.3 RESOURCE PROTECTION CONTROL FLOW MECHANISMS:

Resource protection is a de-facto requirement that must be advocated by every enterprise, organisation, and government entity. The importance of this requirement is further escalated when the entities are performing transactions in an online environment. Information security has been long considered as a crucial factor for e-commerce transactions. It is important to note that lack of sufficient security protection may limit the expansion of e-commerce technology. However, although several e-commerce security mechanisms have been proposed and debated over a number of years, current internet technology still poses a number of incidents pertinent to the loss of information, unauthorized use of resources, and information hacking.



These incidents generate an excruciating cost for the ranging from the loss of revenue to the damage of their reputation. A recent survey shows that the average cost resulted from the worst incident at about £280k - £690k per incident for a large organisation and £27.5k - £55k per incident for a small and medium organisation. Similarly, Digital Ecosystem (DE) faces the identical issues due to its open environment where information and resources are exchanged over the network. With possibly thousands of Small and Medium Enterprises (SMEs) that form series of communities in a DE environment, protecting enterprise resources and acknowledging which entities are trusted to access the resources become extensive tasks for each enterprise. While ensuring security protection is all about upholding the confidentiality, integrity, availability and non-repudiation of information, it is evident that the most consistent and effective way to ensure the preservation of these security properties is through the implementation of authentication, authorisation, encryption, and access control mechanisms. Additionally, the provision of an efficient mechanism to measure the trustworthiness of entities will further strengthen the information and resource protection. While authentication ensures only the right entities that are allowed to consume the resources, authorisation restricts the access over multiple hosted resources based on each entity's privileges. Nevertheless, current research in these areas for a DE environment is still very much limited or not attempted. This research gap further becomes our main motivation to focus our work in. The remainder of this paper is structured as follows:

Section 2 provides an introduction of Digital Ecosystem and its security challenges.

Section 3 provides an overview of our proposed solution.

Section 4 provides an implementation of our proposed solution.

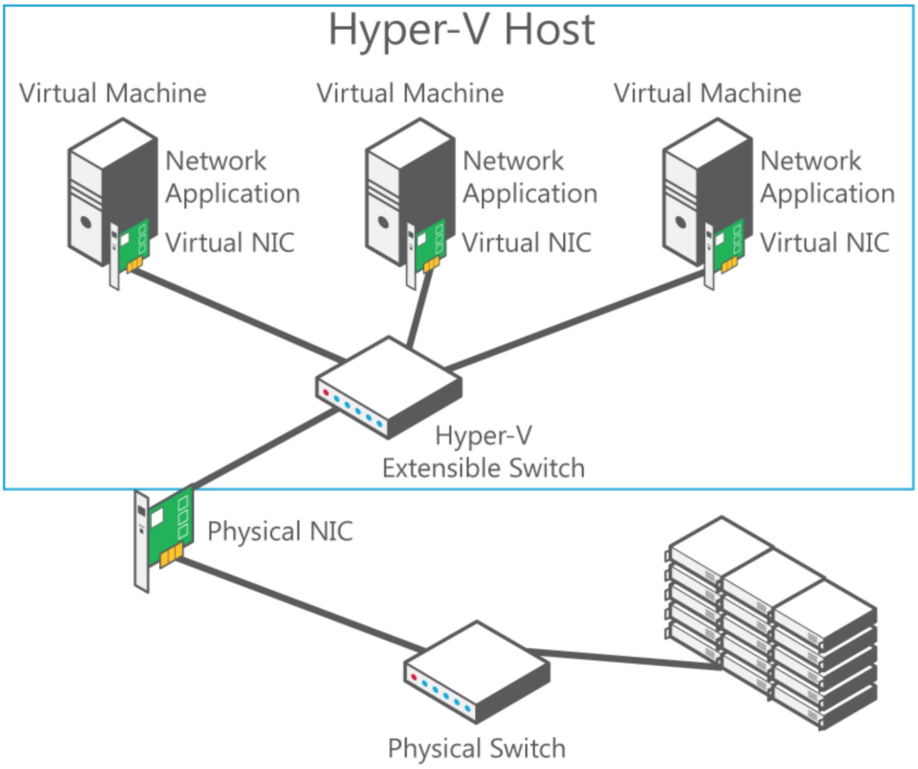
Section 5 presents an security analysis on the proposed solution.

Section 6 which shows the results of performance and scalability testing on our solution. To conclude the paper,

Section 7 summarizes our present work and demonstrates several future works.

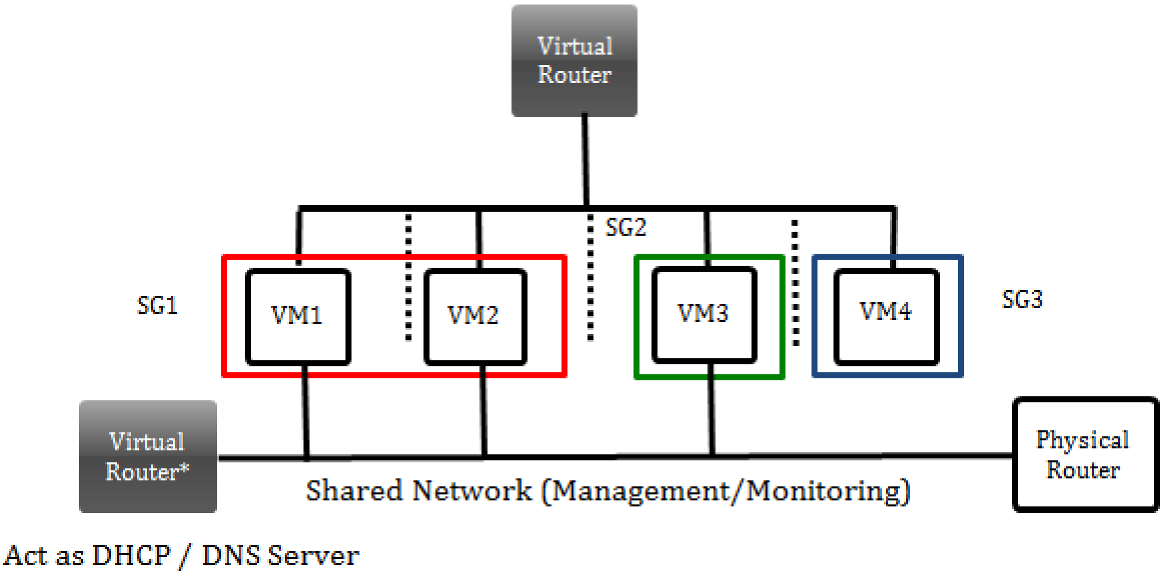
4.4 SECURITY BY ISOLATION

Because of the problems with effectively implementing Security by Correctness approach, people, from the very beginning, has also taken another approach, which is based on isolation. The idea is to split a computer system into smaller pieces and make sure that each piece is separated from the other ones, so that if it gets compromised/malfunctions, then it cannot affect the other entities in the system. Early UNIX's user accounts and separate process address spaces, things that are now present in every modern OS, are examples of Security by Isolation.



Simple as it sound, in practice the isolation approach turned out to be very tricky to implement. One problem is how to partition the system into meaningful pieces and how to set permissions for each piece. The other problem is implementation - e.g. if we take a contemporary consumer OS, like Vista, Linux or Mac OSX, all of them have monolithic kernels, meaning that a simple bug in any of the kernel components (think: hundreds of 3rd party drivers running there), allows to bypass of the isolation mechanisms provided by the kernel to the rest of the system (process separation, ACLs, etc).

Obviously the problem is because the kernels are monolithic. Why not implement Security by Isolation on a kernel level then? Well, I would personally love that approach, but the industry simply took another course and decided that monolithic kernels are better then micro-kernels, because it's easier to write the code for them and (arguably) they offer better performance.



Many believe, including myself, that this landscape can be changed by the virtualization technology. Thin bare-metal hypervisor, like e.g. Xen, can act like a micro kernel and enforce isolation between other components in the system - e.g. we can move drivers into a separate domain and isolate them from the rest of the system. But again there are challenges here on both the design- as well as the implementation-level. For example, we should not put all the drivers into the same domain, as this would provide little improvement in security.

4.5 FUNTIONALITIES IN SOME OPERATING SYSTEM:

Operating system security is provided by gates that users must pass through before entering the operating system environment, and permission matrixes that determine what they are able to do once inside. In some contexts, secure RPC passwords have been referred to as network passwords.

The overall system is composed of four gates and two permission matrixes:

Dialup gate: To access a given operating system environment from the outside through a modem and phone line, you must provide a valid login ID and dial-up password.

Login gate: To enter a given operating system environment you must provide a valid login ID and user password.

Root gate: To gain access to root privileges, you must provide a valid root user password.

Secure RPC gate

In an NIS+ environment running at security level 2 (the default), when you try to use NIS+ services and gain access to NIS+ objects (servers, directories, tables, table entries, and so on) your identity is confirmed by NIS+, using the secure RPC process.

Entering the secure RPC gate requires presentation of a secure RPC password. Your secure RPC password and your login password normally are identical. When that is the case, you are passed through the gate automatically without having to re-enter your password. (In some contexts, secure RPC passwords have been referred to as network passwords. See the Administering NIS+ Credentials section in the AIX 5L™ Version 5.3 Network Information Services (NIS and NIS+) Guide for information about handling two passwords that are not identical.)

A set of credentials is used to automatically pass your requests through the secure RPC gate. The process of generating, presenting, and validating your credentials is called authentication because it confirms who you are and that you have a valid secure RPC password. This authentication process is automatically performed every time you request NIS+ service.

In an NIS+ environment running in NIS-compatibility mode, the protection provided by the secure RPC gate is significantly weakened because everyone has read rights for all NIS+ objects and modify rights for those entries that apply to them regardless of whether or not they have a valid credential (that is, regardless of whether or not the authentication process has confirmed their identity and validated their secure RPC password). Because this situation allows anyone to have read rights for all NIS+ objects and modify rights for those entries that apply to them, an NIS+ network running in compatibility mode is less secure than one running in normal mode. (In secure RPC terminology, any user without a valid credential is considered a member of the nobody class. See Authorization classes for a description of the four classes.)

File and directory matrix

Once you have gained access to an operating system environment, your ability to read, execute, modify, create, and destroy files and directories is governed by the applicable permissions.

NIS+ objects matrix

Once you have been properly authenticated to NIS+, your ability to read, modify, create, and destroy NIS+ objects is governed by the applicable permissions. This process is called NIS+ authorization.

4.6 TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA:

Trusted Computer System Evaluation Criteria (TCSEC) is a United States Government Department of Defence (DoD) standard that sets basic requirements for assessing the effectiveness of computer security controls built into a computer system. The TCSEC was used to evaluate, classify and select computer systems being considered for the processing, storage and retrieval of sensitive or classified information.

TCSEC is divided in four parts: A, B, C, and D, where 'A' describes systems with the highest security and 'D' describes untrusted/untrustworthy systems. Each of these is further subdivided into "classes". Microsoft received "C2" certification for Windows NT. This mean the government certified the system as to conforming to class 2 of division C. Contrast: TCSEC is designed around the concept of trusted employees accessing local systems. It was not designed for todays open Internet access. Hackers do not approach security from the TCSEC point of view. TCSEC doesn't deal with types of threats hackers pose. What this means is that the TCSEC approach is irrelevent when trying to defend your e-commerce site against hackers. However, it is extremely useful in protecting internal systems from internal people. Remember that the biggest threat is from your own internal employees, and that most cybercriminals were convicted for having abused trust placed in them.

UNIT-5

SECURITY SOFTWARE DESIGN:

5.1 A METHODOLOGICAL APPROACH TO SECURITY SOFTWARE DESIG:

Software design is the process of implementing software solutions to one or more set of problems. One of the important parts of software design is the software requirements analysis (SRA). It is a part of the software development process that lists specifications used in software engineering. If the software is "semi-automated" or user centered, software design may involve user experience design yielding a story board to help determine those specifications. If the software is completely automated (meaning no user or user interface), a software design may be as simple as a flow chart or text describing a planned sequence of events. There are also semi-standard methods like Unified Modeling Language and Fundamental modeling concepts. In either case, some documentation of the plan is usually the product of the design. Furthermore, a software design may be platform-independent or platform-specific, depending on the availability of the technology used for the design.

Software design can be considered as creating a solution to a problem in hand with available capabilities. The main difference between Software analysis and design is that the output of a software analysis consist of smaller problems to solve. Also, the analysis should not be very different even if it is designed by different team members or groups. The design focuses on the capabilities, and there can be multiple designs for the same problem depending on the environment that solution will be hosted. They can be operations systems, webpages, mobile or even the new cloud computing paradigm. Sometimes the design depends on the environment that it was developed, whether if it is created from with reliable frameworks or implemented with suitable design patterns).

When designing software, two important factors to consider are its security and usability.

DESIGN CONCEPTS:

The design concepts provide the software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved. They are:

Abstraction - Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically in order to retain only information which is relevant for a particular purpose.

Refinement - It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a step-wise fashion until programming language statements are reached. In each step, one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and Refinement are complementary concepts.

Modularity - Software architecture is divided into components called modules.

Software Architecture - It refers to the overall structure of the software and the ways in which that structure provides conceptual integrity for a system. A good software architecture will yield a good return on investment with respect to the desired outcome of the project, e.g. in terms of performance, quality, schedule and cost.

Control Hierarchy - A program structure that represents the organization of a program component and implies a hierarchy of control.

Structural Partitioning - The program structure can be divided both horizontally and vertically. Horizontal partitions define separate branches of modular hierarchy for each major program function. Vertical partitioning suggests that control and work should be distributed top down in the program structure.

Data Structure - It is a representation of the logical relationship among individual elements of data.

Software Procedure - It focuses on the processing of each modules individually

Information Hiding - Modules should be specified and designed so that information contained within a module is inaccessible to other modules that have no need for such information

DESIGN CONSIDERATIONS:

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. Some of these aspects are:

Compatibility - The software is able to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward-compatible with an older version of itself.

Extensibility - New capabilities can be added to the software without major changes to the underlying architecture.

Fault-tolerance - The software is resistant to and able to recover from component failure.

Maintainability - A measure of how easily bug fixes or functional modifications can be accomplished. High maintainability can be the product of modularity and extensibility.

Modularity - the resulting software comprises well defined, independent components. That leads to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.

Reliability - The software is able to perform a required function under stated conditions for a specified period of time.

Reusability - the software is able to add further features and modification with slight or no modification.

Robustness - The software is able to operate under stress or tolerate unpredictable or invalid input. For example, it can be designed with a resilience to low memory conditions.

Security - The software is able to withstand hostile acts and influences.

Usability - The software user interface must be usable for its target user/audience. Default values for the parameters must be chosen so that they are a good choice for the majority of the users.

MODELING LANGUAGE

A modeling language is any artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of components in the structure. A modeling language can be graphical or textual. Examples of graphical modeling languages for software design are:

i. Business Process Modeling Notation (BPMN) is an example of a Process Modeling language.

ii. EXPRESS and EXPRESS-G (ISO 10303-11) is an international standard general-purpose data modeling language.

iii. Extended Enterprise Modeling Language (EEML) is commonly used for business process modeling across a number of layers.

iv. Flowchart is a schematic representation of an algorithm or a step-wise process,

v. Fundamental Modeling Concepts (FMC) modeling language for software-intensive systems.

vi. IDEF is a family of modeling languages, the most notable of which include IDEF0 for functional modeling, IDEF1X for information modeling, and IDEF5 for modeling ontologies.

vii. Jackson Structured Programming (JSP) is a method for structured programming based on correspondences between data stream structure and program structure

viii. LePUS3 is an object-oriented visual Design Description Language and a formal specification language that is suitable primarily for modelling large object-oriented (Java, C++, C#) programs and design patterns.

ix. Unified Modeling Language (UML) is a general modeling language to describe software both structurally and behaviorally. It has a graphical notation and allows for extension with a Profile (UML).

x. Alloy (specification language) is a general purpose specification language for expressing complex structural constraints and behavior in a software system. It provides a concise language based on first-order relational logic.

xi. Systems Modeling Language (SysML) is a new general-purpose modeling language for systems engineering.

DESIGN PATTERNS

A software designer or architect may identify a design problem which has been solved by others before. A template or pattern describing a solution to a common problem is known as a design pattern. The reuse of such patterns can speed up the software development process, having been tested and proven in the past.

USAGE

Software design documentation may be reviewed or presented to allow constraints, specifications and even requirements to be adjusted prior to computer programming. Redesign may occur after review of a programmed simulation or prototype. It is possible to design software in the process of programming, without a plan or requirement analysis,[3] but for more complex projects this would not be considered a professional approach. A separate design prior to programming allows for multidisciplinary designers and Subject Matter Experts (SMEs) to collaborate with highly skilled programmers for software that is both useful and technically sound.

SECURE OPERATING SYSTEM DESIGN

Most modern information computer systems provide concurrent execution of multiple applications in a single physical computing hardware (which may contain multiple processing units). Within such a multitasking, time-sharing environment, individual application jobs share the same resources of the system, e.g., CPU, memory, disk, and I/O devices, under the control of the operating system. In order to protect the execution of individual application jobs from possible interference and attack of other jobs, most contemporary operating systems implement some abstract property of containment, such as process (or task) and TCB (Task Control Block), virtual memory space, file, port, and IPC (Inter Process Communication), etc. An application is controlled that only given resources (e.g., file, process, I/O, IPC) it can access, and given operations (e.g., execution or read-only) it can perform. However, the limited containment supported by most commercial operating systems (MS WIndows, various flavors of Unix, etc) bases access decisions only on user identity and ownership without considering additional security-relevant criteria such as the operation and trustworthiness of programs, the role of the user, and the sensitivity or integrity of the data. As long as users or applications have complete discretion over objects, it will not be possible to control data flows or enforce a system-wide security policy. Because of such weakness of current operating systems, it is rather easy to breach the security of an entire system once an application has been compromised, e.g., by a buffer overflow attack. Some examples of potential exploits from a compromised application are:

i. Use of unprotected system resources illegitimately. For example, a worm
ii. program launches attack via emails to all targets in the address book of a
iii. user after it gets control in a user account.
iv. Subversion of application enforced protection through the control of
v. underneath system. For example, to deface a Web site by gaining the
vi. control of the Web server of the site, say changing a virtual directory in
vii. Microsoft IIS.
viii. Gain direct access to protected system resources by misusing privileges.
ix. For example, a compromised "sendmail" program running as root on a
x. standard Unix OS will result in super user privileges for the attacker and
xi. uncontrolled accesses to all system resources.
xii. Furnish of bogus security decision-making information. For example,
xiii. spoof of a file handle of Sun's NFS may easily give remote attackers
xiv. gaining access to files on the remote file server.

It is not possible to protect against malicious code of an application using existing mechanisms of most commercial operating systems because a program running under the name of a user receives all of the privileges associated with that user. Moreover, the access controls supported by the operating systems are so coarse – only two categories of users: either completely trusted super users (root) or completely un-trusted ordinary users. As the result, most system services and privileged applications in such systems have to run under root privileges that far exceed what they really needed. A compromise in any of these programs would be exploited to obtain complete system control.

DBMS DESIGN SECURITY PACKAGE

The result of database design is a plan for the construction of a database that captures some of the essential features of the proposed database, but omits a lot of less important detail. These plans often take the form of data models, and database design can be learned as the art of constructing a certain kind of data model.

Most databases that capture and manage semi-permanent data operate under the control of a database management system (DBMS). Prominent DBMS products are Microsoft's SQL Server, Oracle RDBMS, and IBM's DB2. There are dozens of others. Many of the questions and answers you'll find under this tag relate to one of these DBMS products, but some design issues are DBMS independent.

The amount of preparation and education you'll need before building your first successful database varies widely depending on many factors. Among other factors, it depends on how ambitious your database project is and on what prior experience you bring to bear on the project. Very experienced programmers sometimes underestimate the amount of material there is to learn about database design.

Sometimes programmers learn well by trial and error, or by postponing formal learning until their second or third project. Other times, database design neophytes make design decisions that lead into pitfalls that are very difficult to reverse.

There are many ways to measure the quality of a database design. Programmers building their first database are often primarily concerned with performance. There's no question that performance is important. A bad design can easily result in database operations that take ten to a hundred times as much time as they should.

But don't let performance issues blind you to other aspects of good design. In particular, future proofing of a database is enormously important. Failure to do this can result in a database that traps its users at the first level and prevents their data from evolving as their needs evolve.

Another aspect involves separating out the hidden features of a database (sometimes called physical design) from the public features visible across the application interface (sometimes called logical design). A neat separation of these features can result a database that can be tweaked and tuned quite a bit with no changes to application code. A poor separation of these features can result in a database that makes a nightmare out of application development or database administration.

Another consideration is whether the proposed database will be embedded within a single application, or whether it will be an information hub that serves the needs of multiple applications. Some design decisions will be made very differently in these two cases.

Yet another consideration is whether the application is going to perform all data management functions on behalf of its clients, or whether custodial responsibility for the database and its data is going to be vested in one or more DBAs (database administrators)

UNIT-6

STATISTICAL DATABASE PROTECTION & INTRUSION DETECTION SYSTEM

INTRODUCTION

STATISTICS CONCEPTS AND DEFINATIONS:

Statistics - a set of concepts, rules, and procedures that help us to:

- Organize numerical information in the form of tables, graphs, and charts;
- Understand statistical techniques underlying decisions that affect our lives and well-being; and
- Make informed decisions.

Statistical databases are databases containing statistical information. Such databases are normally released by national statistical institutes but, on occasion, they can also be released by healthcare authorities (epidemiology) or by private organizations (e.g. consumer surveys). Statistical databases typically come in three formats:

• Tabular data, that is, tables with counts or magnitudes, which are the classical output of official statistics;

• Queryable databases, that is, on-line databases to which the user can submit statistical queries (sums, averages, etc.);

• Micro data , that is, files where each record contains information on an individual (a citizen or a company).

The peculiarity of statistical databases is that they should provide useful statistical information, but they should not reveal private information on the individuals they refer to (respondents). Indeed, supplying data to national statistical institutes is compulsory in most countries but, in return, those institutes commit to preserving the privacy of respondents. Inference control in statistical databases, also known as Statistical Disclosure Control (SDC), is a discipline that seeks to protect data in statistical databases so that they can be published without revealing confidential information that can be linked to specific individuals among those to which the data correspond. SDC is applied to protect respondent privacy in areas such as official statistics, health statistics, e-commerce (sharing of consumer data), etc. Since data protection ultimately means data modification, the challenge for SDC is to achieve protection with minimum loss of the accuracy sought by database users. In, a distinction is made between SDC and other technologies for database privacy, like privacy-preserving data mining (PPDM) or private information retrieval (PIR): what makes the difference between those technologies is whose privacy they seek. While SDC is aimed at respondent privacy, the primary goal of PPDM is to protect owner privacy when several database owners wish to co-operate in joint analyses across their databases without giving away their original data to each other. On its side, the primary goal of PIR is user privacy, that is, to allow the user of a database to retrieve some information item without the database exactly knowing which item was recovered. The literature on SDC started in the 1970s, with the seminal contribution by Dalenius in the statistical community and the works by Schl¨rer and Denning in the database community. The 1980s saw moderate activity in this field. An excellent survey of the state of the art at the end of the 1980s is in the 1990s, there was renewed interest in the statistical community and the discipline was further developed under the names of statistical disclosure control in Europe and statistical disclosure limitation in America. Subsequent evolution has resulted in at least three clearly differentiated sub disciplines:

- Tabular data protection. The goal here is to publish static aggregate information, i.e. tables, in such a way that no confidential information on specific individuals among those to which the table refers can be inferred. See for a conceptual survey.

- Queryable databases. The aggregate information obtained by a user as a result of successive queries should not allow him to infer information on specific individuals. Since the late 70s, this has been known to be a difficult problem, subject to the tracker attack. SDC strategies here include perturbation, query restriction and camouflage (providing interval answers rather than exact answers).

- Microdata protection. It is only recently that data collectors (statistical agencies and the like) have been persuaded to publish microdata. Therefore, microdata protection is the youngest sub discipline and is experiencing continuous evolution in the last years. Its purpose is to mask the original microdata so that the masked microdata are still analytically useful but cannot be linked to the original respondents.

There are several areas of application of SDC techniques, which include but are not limited to the following:

- Official statistics. agencies to guarantee statistical confidentiality when they release data collected from citizens or companies. This justifies the research on SDC undertaken by several countries, among them the European Union (e.g. the CASC project) and the United States.

- Health information. This is one of the most sensitive areas regarding privacy. For example, in the U. S., the Privacy Rule of the Health Insurance Portability and Accountability Act (HIPAA) requires the strict regulation of protected health information for use in medical research. In most western countries, the situation is similar.

- E-commerce. Electronic commerce results in the automated collection of large amounts of consumer data. This wealth of information is very useful to companies, which are often interested in sharing it with their subsidiaries or partners. Such consumer information transfer should not result in public profiling of individuals and is subject to strict regulation.

6.2 TYPES OF ATTACKS INFRENCE CONTROL EVALUTION CRETERIA FOR CONYROL COMPARISON:

Inference attacks are notoriously hard to mitigate due to the fact that some data always needs to be made available to legitimate sources. It's difficult to prevent a determined individual from connecting available non-sensitive data and making inferences about more sensitive data. With more databases reachable from the web, this opens numerous opportunities for hackers to gain knowledge about sensitive or confidential data which they should not have.

Database inference is not easily categorized in any other group of information security attacks. This is due to the fact that an inference attack leverages the human mind, or similar logic systems, in order to obtain data that may be considered secure in the traditional sense There are many definitions of what an inference is, but in the context of database security it is defined as the act or process of deriving sensitive information from premises known or assumed to be true. The _premises known or assumed to be true_ may be freely/publicly available information or information gleaned through other methods.

DATABASE INFERENCE FOR DATA MINING:

When an attacker attempts inferencing they generally have some idea what they are looking for. They may start out with that knowledge, or they may not. How would they initially know what they should be looking for? Data Mining is a technique used to gather data and find frequent patterns, find associations between data and build rules for those associations and patterns. For example, data mining techniques may determine that there are a lot of references to particular words or phrases in a group of documents stored in a database. It may also determine that there are associations which can be predicted (if a group of data contains items a and b, it is highly likely to also contain item c). These predictions can be formed into rules which can be applied using inferencing techniques to infer missing or restricted data. Data mining can utilize any collected data, although generally publicly available (via the web) sources are used. Data may also be found accidentally, or through social engineering.

n-item k-percent rule violations:

This rule applies to statistical data sets where only aggregate queries have been allowed. It says that whenever a query is made some number of items (N) should not represent greater than a certain percentage (k) of the result reported. This is to ensure that a where clause isn't added to an aggregate query which reduces the rows calculated to few enough to to infer speci_c data items. The most obvious case is where 1-item represents 100-percent of the result. In other words a where clause has been tailored to return only a single row value which represents the entire result, therefor defeating the requirement that only aggregate queries be allowed.

Unencrypted Index

While secure databases are often encrypted, the indexes fre- quently remain unencrypted for quicker access. Indexes are used to make searches and certain queries run more quickly. Encrypting them defeats this purpose to some extent and there- fore frequently the index are left in plain text. Data from un- encrypted indexes can be used to piece together closely related data just by noting table names and keys.

What's at Stake?

Database inference is an information security issue. Whenever we talk about information security we think about the CIA model (Confidentiality, Integrity, and Availability). Database inferencing is all about compromising confidentiality. The end result of a successful inference attack is equivalent to a leak of sensitive information. Even if actual information is not leaked, certain statistics about that information can provide enough information to make inferences which may still constitute a legitimate breach. Methods of Attack There are several different methods used for effecting inference attacks. They can be used individually or more commonly (and most effectively) in conjunction with each other. Out of Channel Attacks _Out of Channel_ refers to using information from outside sources to attack the target database. Most inference attacks are affected using at least some out of channel data, but it's not necessary. Extensive data mining of numerous publicly accessible information sources and using that data to infer data in a secured database is a good example. Out of channel attacks are extremely difficult to guard against as frequently the data is out of the control of the target. The web makes all types of information easily available and search-able. It's not always possible or feasible to remove these sources of information.

Direct Attacks

These are attacks directly on the target database. They seek to find sensitive information directly with queries that yield only a few records. These are the easiest to detect and deny. MAC and DAC methods can mitigate these types of attacks by ensuring data is properly classified. Similarly, triggers can be written to ensure that queries conform to security policy standards. Direct attacks are most effective when database security is lax or systems have been misconfigured.

Indirect Attacks

Indirect attacks seek to infer the final result based on a number of intermediate results. Intermediate result may be obtained by aggregate (Sum, Count, Median, etc) or set theory. A number of complex and surprisingly effective techniques can be used. Intersections of sets can be examined. With statistical databases linear systems of equations can be utilized to solve for missing (sensitive) data values.

Inferencing Categories

Logical Inferences

Uses association rules such as those gleaned from data mining to make logical assumptions about data. If a, b, c, d and a ,b ,e , d then probably a, b, f _ d. Logical inferences are most commonly used to make associations between textual data. Techniques borrowed from data mining such as apriori and clustering. These generally fall under the category of direct attacks, but can also be considered indirect when more complex methods are used.

Statistical Inferences

Takes aggregate data and uses math/statistics to derive data pertaining to individuals in the data set. Statistical inferencing is generally applied to numerical data sets but can be extended to use with textual data. Textual data can easily be enumerated or represented as frequencies or counts. The same statistical methods can then be used to derive associations. These generally fall under the category of indirect attacks since result are based on a combination (sometimes quite complex) of intermediate results frequently based on aggregate data.

UNIT-7

MODELS FOR THE PROTECTION OF NEW GENERATION DATABASE-1

1. A MODEL FOR FRAME BASED SYSTEM:

Frame based systems use entities like frames and their properties as a modeling primitive. The central modeling primitive is a frame together with slots. These slots are applicable only to the frames they are defined for. Value restriction (facets) can be defined for each attribute. A frame provides a context for modeling one aspect of a domain. An important part of frame-based languages is the possibility of inheritance between frames. The inheritance allows inheriting attributes together with restrictions on them. Knowledge base then consists from instances (objects) of these frames.

An example of the usage of the frame-based model is Open Knowledge Base Connectivity (OKBC) that defines API for accessing knowledge representation systems. It defines most of the concepts found in frame-based systems, object databases and relational databases. The OKBC API is defined in language independent fashion, and implementations exist for Common Lisp, Java, and C. The OKBC API provides operations for manipulating knowledge expressed in an implicit representation formalism called the OKBC Knowledge Model. The conceptualization in OKBC Knowledge Model is based on frames, slots, facets, instances, types, and constants. This knowledge model supports an object- oriented representation of knowledge and provides a set of representational constructs and thus can serve as an interlingua for knowledge sharing and translation. The OKBC Knowledge Model includes constants, frames, slots, facets, classes, individuals, and knowledge bases. For precise description of the model, the KIF language (see section about KIF) is used.

The OKBC knowledge model assumes a universe of discourse consisting of all entities about which knowledge is to be expressed. In every domain of discourse it is assumed that all constants of the following basic types are always defined: integers, floating point numbers, strings, symbols, lists, classes. It is also assumed that the logical constants true and false are included in every domain of discourse. Classes are sets of entities, and all sets of entities are considered to be classes.

A frame is a primitive object that represents an entity in the domain of discourse. A frame is called class frame when it represents a class, and is called individual frame when it represents an individual. A frame has associated with it a set of slots that have associated a set of slot values. A slot has associated a set of facets that put some restrictions on slot values. Slots and slot values can be again any entities in the domain of discourse, including frames. A class is a set of entities, that are instances of that class (one entity can be instance of multiple classes). A class is a type for those entities. Entities that are not classes are referred to as individuals. Class frames may have associated a template slots and template facets that are considered to be used in instances of subclasses of that class. Default values can be also defined. Each slot or facet may contain multiple values. There are three collection types: set, bag (unordered, multiple occurrences permitted), and list (ordered bag). A knowledge base (KB) is a collection of classes, individuals, frames, slots, slot values, facets, facet values, frame-slot associations, and frame-slot-facet associations. KBs are considered to be entities of the universe of discourse and are represented by frames. There are defined standard classes, facets, and slots with specified names and semantics expressing frequently used entities

2. <u>A MODEL FOR THE PROTECTION OF OBJECT ORIENTED SYSTEM:</u>

Much attention has recently been directed towards the development of object-oriented programming and object-oriented systems [COX86]. Its notion is a natural consequence of modeling any entity in the real world as an object. Thus, an object could be as simple as an integer or as complex as an automobile. Object-oriented para- digm has been extended to model database systems also by providing support for persistence and schema management among others. Simultaneously, attention is also being directed to the design and development of secure database systems, including multilevel se- cure database management systems (MLS/DBMS). Such systems are recognized as crucial for the secure operation of military appli- **cations**. In a multilevel secure database management system, users cleared to different levels an be expected to share a database consisting of data at a variety of sensitivity levels. Until recently MLS/ DBMS research has been focused on the relational model of Codd [CODD70]. The relational model, although rich in theory, lacks the flexibility and power of representation of the object model. There- fore, it is expected that many new generation applications will gradually utilize the object model [LU87, KONA88].

Security Policy

It is assumed that the multilevel secure object-oriented database management system (MLS/ODBMS) is hosted on a TCB (Trusted Computing Base), TCB is the solution proposed for multilevel security in operating systems. The security policy commonly used by most TCBs is the Bell and LaPadula security policy [BELL75], This policy consists of the following two propenies:

l. Simple property: A subject has read access to an object if the subject’s security clearance dominates the sensitivity level of the object (e. g., a secret subject can read either secret or unclassified objects).

2. *-property: A subject has write access to an object if the subject's clearance level is dominated by the security level of the object (e.g., an unclassiñed subject can write into an unclassiñed or secret object). A user may assign security levels to classes, instances, instance variables, and methods. These security levels are assigned via the mandatory security constraints. After the constraints are defined and stored in the constraint database, whenever a user requests to create a class, the schema manager component ofthe DBMS, which is responsible for managing the constraints, will examine the security constraints and determine the security level of the class. In addition, the schema manager may determine that some additional classes should also be created. Ultimately, the security levels are assigned to only the classes. The instances, instance variables, and methods associated with a class will get the security level of the class. That is, ifa class is assigned say a secret security level, then all of its instances, methods, and instance variables are secret.

UNIT-8

MODELS FOR THE PROTECTION OF NEW GENERATION DATABASE SYSTEMS-2

THE ORION DATA MODE:

The ORION authorization model permits access to data on the basis of explicit authorizations provided to each group of users. These authorizations are classified as positive authorizations because they specifically allow a user access to an object. Similarly, a negative authorization is used to specifically deny a user access to an object. The placement of an individual into one or more groups is based on the role that the individual plays in the organization. In addition to the positive authorizations that are provided to users within each group, there are a variety of implicit authorizations that may be granted based on the relationships between subjects and access modes.

Orion 2.0 has two major kinds of attributes uncertain and certain. A database table T is defined by a probabilistic schema consisting of a schema and dependency information. The schema is similar to the regular relational schema and species the names and data types of the table attributes (both certain and uncertain). The dependency information identifies the attributes in that are jointly distributed (i.e. correlated). For each dependent set of attributes in the model maintains a history.

Attributes: The uncertainty in many applications can be expressed using standard distributions. Orion has built-in support for commonly used continuous (e.g. Gaussian, Uniform) and discrete (e.g. Binomial, Poisson) distributions. These uncertain values are processed symbolically in the database. When the underlying data cannot be represented using standard distributions, Orion automatically converts them to approximate distributions, including histograms and discrete sampling.

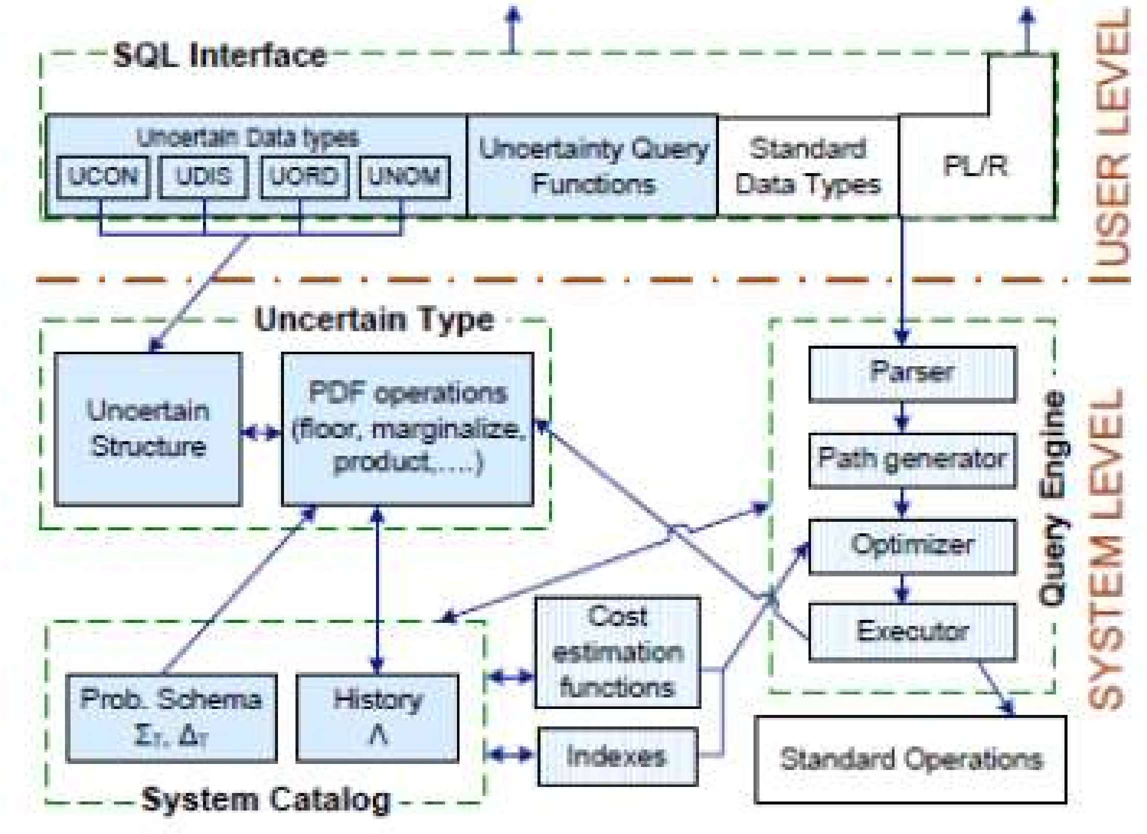


Figure 1: Orion 2.0 Architecture

Correlations and missing data: Correlated attributes in a table T(given by ΔT) are represented by a single joint distribution. An important feature of Orion is its support for partial. A partial is a distribution that sums (or integrates) to less than 1. This feature allows us to represent missing tuples. If the joint of a tuple (obtained by multiplying the individual of attributes) sums up to x, then 1-x is the probability that the tuple does not exist in the database.

Historical dependencies: In addition to the for each dependent group of uncertain attributes present in ΔT , we store its history ^. While dependency information expresses intra-tuple dependencies at the schema level, history captures the inter-tuple dependencies at the instance level. The history of a given set of uncertain attributes denotes the attribute sets from which it is derived and is used while performing operations on the tuples to ensure that the results are consistent with PWS.

Operations: Correct evaluation of select-project join queries under PWS reduces to three fundamental operations on floor, marginalize, and product. These operations use the information maintained by dependency sets in ΔT and histories into detect any correlations and handle them appropriately. The standard relational operations remain unchanged for the certain attributes in the database.

RETISS system

A real-time security system (RETISS) for threat detection is described, pointing out security violations in the target system under control. RETISS is based on the hypothesis that a correlation exists between anomalous user behavior and threats. Security rules have been enforced to express this correlation and to detect and evaluate the probability of a given threat, based on the level of danger of the occurrences of the anomalies symptomatic for the threat. Levels of danger of all the anomalies are then fuzzy combined to express the probability of the threat. RETISS is independent of any particular system and application environment. Moreover, RETISS runs on a machine different from that of the target system in order to be protected against attacks from users of the target system