# Database Security Control Measures

We follow **Elmasrı R., Navathe S.B., Fundamentals of Database Systems, 5th Ed., Pearson, 2007, Chapter 23**

There are four main control measures that are used to provide security of data in databases:

* Access control
* Inference control
* Flow control
* Data encryption

Access control we have already considered and we shall extend it. Inference control measures are used to prevent extraction (inference) of private information from publicly available statistical databases such as salary summary in universities’ departments. Flow control prevents information from flowing in such a way that it reaches unauthorized users. Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called covert channels. A final control measure is data encryption. We have also touched this issue.

# Database Security and the DBA

The database administrator (DBA) is the central authority for managing a database system. The DBA’s responsibilities include account creation (access control as a whole), granting/revoking privileges to/from users (discretionary authorization control) and classifying users and data in accordance with the policy of the organization (mandatory authorization control). The DBA has a DBA account, sometimes called a system or super-user account, which provides powerful capabilities that are not made available to regular database accounts and users.

# Access Protection, User Accounts, and Database Audits

Each valid user gets from DBA an account number and password, which are used for user authentication when he logs in the system. For each login session, the DBMS can record the user’s account number and associate it with the terminal from which the user logged in. All operations applied from the terminal are attributed to the user’s account until the user logs off. It is particularly important to keep track of update operations that are applied to the database so that, if the database is tampered with, the DBA can understand which user did the tampering. If any tampering of the database is suspected, a database audit is performed, which consists of reviewing the

# Access Protection, User Accounts, and Database Audits (Cont 1)

log file (audit file) to examine all accesses and operations applied to the database during a certain time period.

# Discretionary Access Control Based on Granting and Revoking Privileges

There are two levels for assigning privileges to use the database system:

* The account level. At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database (e.g., CREATE SCHEMA and CREATE TABLE privileges)
* The relation (table) level. At this level, the DBA can control the privilege to access each individual relation or view in the database (user access control matrix, where the rows represent subjects (users, accounts, programs) and the columns represent objects (relations, records, columns, views, operations)

To control the granting and revoking of relation privileges, each relation R in a database is assigned an owner account, which is typically the account that was used when the relation was created. The owner of the relation is given all privileges on that relation. The owner account holder can pass privileges on any of the owned relations to other users by granting privileges to their accounts. In SQL the following types of privilege can be granted on each individual relation R:

* Select privilege
* Modify privilege
* References privilege

To create a view, the account must have SELECT privilege on all relations in the view definition.

The mechanism of views is an important discretionary authorization mechanism in its own right. It can be used to restrict access to particular fields and tuples.

Privileges can be propagated using the GRANT OPTION, for example,

GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION

User A3 can now propagate the privilege to other accounts by using GRANT.

# Discretionary Access Control Based on Granting and Revoking Privileges (Cont 1)

GRANT SELECT ON EMPLOYEE TO A4

does not give propagation right to A4.

REVOKE SELECT ON EMPLOYEE FROM A3

revokes select privilege on Employee from A3.

Limits on propagation of privileges can be implemented in two ways. Limiting horizontal propagation to an integer number *i* means that an account B given the GRANT OPTION can grant the privilege to at most *i* other accounts. Vertical propagation is more complicated; it limits the depth of the granting of privileges. Granting a privilege with a vertical propagation of zero is equivalent to granting the privilege with no GRANT OPTION. If account A grants a privilege to account B with the vertical propagation set to an integer number *j>0*, this means that the account B has the GRANT OPTION on that privilege, but B can grant the privilege to other accounts only with vertical propagation less than *j*. In effect, vertical propagation limits the sequence of grant options that can be given from one account to the next based on a single original grant of the privilege.

# Mandatory Access Control

In addition to discretionary access control, in many applications, it is necessary to classify data and users based on security classes. This is known as mandatory access control. The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each subject and object into one of the security classifications such as TS (top-secret), S (secret), C (confidential), or U (unclassified), with TS>=S>=C>=U. Two restrictions are enforced on data access: no read-up (simple security property), and no write-down (star property, \*-property). These we have considered earlier also.

To incorporate multilevel security into the relational database model, it is common to consider attribute values and tuples as data objects. Hence, each attribute A is associated with a classification attribute C in the schema, and each attribute value in a tuple is associated with a corresponding security classification. In addition, in some models, a tuple classification attribute TC is added to the relation attributes to provide a classification for each tuple as a whole. Hence, a multilevel relation schema R with n attributes would be represented as

# Mandatory Access Control (Cont 1)

R(A1,C1,A2,C2,..,An,Cn,TC)

where each Ci represents the classification attribute associated with attribute Ai.

The value of the TC attribute in each tuple t – which is the highest of all attribute classification values within t – provides a general classification for the tuple itself, whereas each Ci provides a finer security classification for each attribute value within the tuple. The apparent key of a multilevel relation is the set of attributes that would have formed the primary key in a regular (single-level) relation. A multilevel relation will appear to contain different data to subjects (users) with different clearance levels. In some cases, it is possible to store a single tuple in the relation at a higher classification level and produce the corresponding tuples at a lower-level classification through a process known as filtering. In other cases, it is necessary to store two or more classification tuples at different classification levels with the same value for the apparent key. This leads to the concept of poly-instantiation, where several tuples can have the same apparent key value but have different attribute values for users at different classification levels.

We illustrate these concepts with the simple example of a multilevel relation shown in Figure 23.2 (a), where we display the classification attribute values next to each attribute’s value. Assume that the Name attribute is the apparent key, and consider the query

SELECT \* FROM EMPLOYEE.

A user with security clearance S would see the same relation shown in Figure 23.2(a), since all tuple classifications are less than or equal to S.

# Mandatory Access Control (Cont 2)

(a) Employee – the original tuples

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Salary | JobPerformance | TC |
| Smith U | 40000 C | Fair S | S |
| Brown C | 80000 S | Good C | S |

(b) Employee – after filtering for classification C users

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Salary | JobPerformance | TC |
| Smith U | 40000 C | NULL C | C |
| Brown C | NULL C | Good C | C |

(c) Employee - after filtering for classification U users

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Salary | JobPerformance | TC |
| Smith U | NULL U | NULL U | U |

(d) Employee – poly-instantiation of the Smith tuple

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Salary | JobPerformance | TC |
| Smith U | 40000 C | Fair S | S |
| Smith U | 40000 C | Excellent C | C |
| Brown C | 80000 S | Good C | S |

Figure 23.2 A multilevel relation to illustrate multilevel security

However, a user with security clearance C would not be allowed to see values for salary of ‘Brown’ and JobPerformance of ‘Smith’, since they have higher classification. The tuples would be filtered to appear as shown in Figure 23.2(b), with Salary and JobPerformance appearing as null. For user with security clearance U, the filtering allows only the Name attribute of ‘Smith’ to appear, with all other attributes appearing as null (Figure 23.2(c)). Thus, filtering introduces null values for attribute values whose security classification is higher than the user’s security clearance.

In general, the **entity integrity rule** for multilevel relations states that all attributes that are the members of the apparent key must not be null and must have the same security classification within each individual tuple. Additionally, all other attribute values in the tuple must have a security classification greater or equal to that of the apparent key. This constraint ensures that a user can see the key if the user is permitted to see any part of the tuple. Other integrity rules, called **null integrity** and **inter-instance** **integrity**, informally ensures that if a tuple value at some security level can be filtered (derived) from a higher-classified tuple, then it is sufficient to store the higher classified tuple in the multilevel relation.

# Mandatory Access Control (Cont 3)

To illustrate poly-instantiation further, suppose that a user with security clearance C tries to update the value of JobPerformance of ‘Smith’ in Figure 23.2 to ‘Excellent’; this corresponds to the following SQL update being used:

UPDATE EMPLOYEE

SET JobPerformance=’Excellent’

WHERE Name=’Smith’

Since the view provided to users with security clearance C (see Figure 23.2(b)) permits such an update, the system should not reject it; otherwise, the user could infer that some non-null value exists for the JobPerformance attribute of ‘Smith’ rather than the null value that appears. This is an example of inferring information through what is known as a covert channel, which should not be permitted in highly secure systems. However, the user should not be allowed to overwrite the existing value of JobPerformance at the higher classification level. The solution is to create a poly-instantiation for the ’Smith’ tuple at the lower classification level C, as shown in Figure 23.2(d). This is necessary since the new tuple cannot be filtered from the existing tuple at classification S.

The basic update operations INSERT, DELETE, UPDATE must be modified to handle this and similar situations, but we do not consider these issues here.

# Comparing Discretionary Access Control (DAC) and Mandatory Access Control (MAC)

DAC policies are characterized by a high degree of flexibility, which makes them suitable for a large variety of application domains. The main drawback of DAC models is their vulnerability to malicious attacks such as Trojan horses embedded in application programs (we have already seen it previously). MAC policies ensure a higher degree of protection – in a way, that prevents any illegal flow of information. The drawback of MAC is that it is too rigid: it requires a strict classification of subjects and objects into security levels, and therefore, they are applicable to very few environments, as military applications. In many practical situations, DAC is preferred because it offers a better trade-off between security and applicability.

# Role-Based Access Control (RBAC)

RBAC emerged in 1990s as a proven technology for managing and enforcing security in large-scale enterprise-wide systems. Its basic notion is that permissions are associated with roles, and users are assigned to appropriate roles. Roles can be created using the CREATE ROLE and DESTROY ROLE commands. The GRANT and REVOKE commands discussed previously can then be used to assign and revoke privileges from roles.

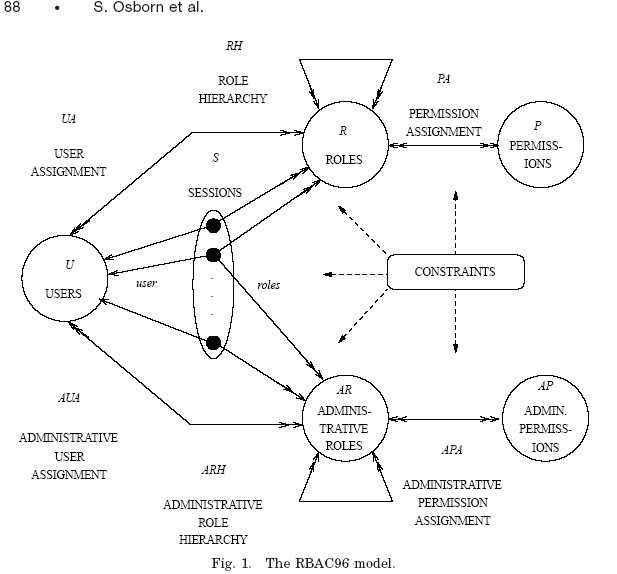
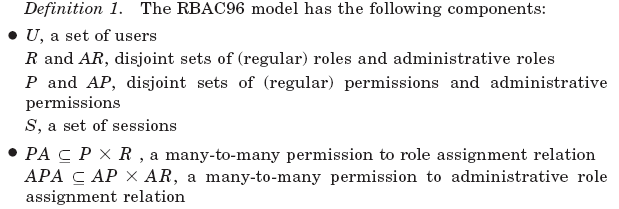
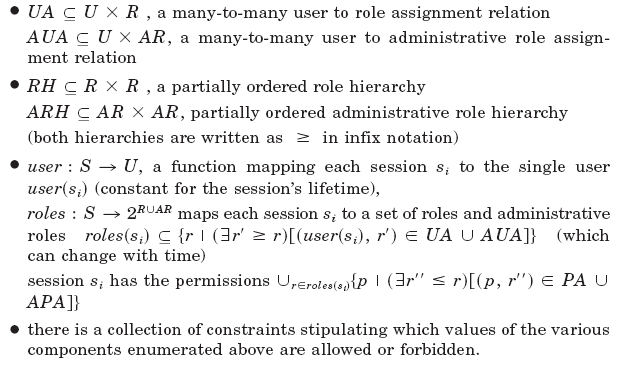


Figure 1 is taken from **[Osborn S., Sandhu R., Munawer Q., Configuring role-based access control to enforce mandatory and discretionary access control policies. – ACM Transactions on Information and System Security, Vol. 3, No. 2, May 2000, 85-106,** [**https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE552/CMPE552%20Fall2021/OsbornACMTISS00.pdf**](https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE552/CMPE552%20Fall2021/OsbornACMTISS00.pdf) **]**

# Role-Based Access Control (Cont 1)





RBAC appears to be a viable alternative to traditional DAC and MAC. It ensures that only authorized users are given access to certain data or resources. Users create sessions during which they may activate a subset of roles to which they belong. Each session can be assigned to many roles, but it maps to one single user or a subject only. Many DBMS have allowed the concept of roles, where privileges can be assigned to roles (e.g., Oracle).

Role hierarchy in RBAC is a natural way to organize roles to reflect the organization’s lines of authority and responsibility. By convention, junior roles at the bottom are connected to progressively senior roles as one moves up hierarchy. The hierarchy diagrams are partial orders, so they are reflexive, transitive, and anti-symmetric:

# Role-Based Access Control (Cont 2)



Another important consideration in RBAC systems is the possible temporal constraints that may exist on roles, such as the time and duration of role activations, and triggering of a role by an activation of another role. Using an RBAC model is a highly desirable goal for addressing the key security requirements of Web-based applications. Roles can be assigned to workflow tasks so that a user with any of the roles related to a task may be authorized to execute it and may play a certain role for certain duration only.

## SQL Injection Attack (Security in computing / C. P. Pfleeger, S. L.Pfleeger, J.Margulies.—Fifth edition, ISBN 978-0-13-408504-3,2015, Pearson Education, p. 290)

Often queries are composed through a browser and transmitted to the database server supporting the web page. A bank might have an application that allows a user to download all transactions involving the user’s account. After the application identifies and authenticates the user, it might compose a query for the user on the order of

QUERY = “SELECT \* FROM trans WHERE acct=’”+ acctNum + ”’;”

Because the communication is between an application running on a browser and the web server, the query is encoded within a long URL string

[http://www.mybank.com?QUERY=SELECT%20\*%20FROM%20trans%20WHERE%20acct=‘2468](http://www.mybank.com?QUERY=SELECT%20*%20FROM%20trans%20WHERE%20acct='2468)’

In this command, the space character has been replaced by its numeric equivalent %20

(because URLs cannot contain spaces), and the browser has substituted ‘2468’ for the

account number variable. The DBMS will parse the string and return records

appropriately. If the user can inject a string into this interchange, the user can force the DBMS to return a set of records. The DBMS evaluates the WHERE clause as a logical expression. If the user enters the account number as

“‘2468’ OR ‘1’=‘1’”

the resulting query becomes

QUERY = “SELECT \* FROM trans WHERE acct=’”+ acctNum + ”’;”

and after account number expansion it becomes

QUERY = “SELECT \* FROM trans WHERE acct=‘2468’OR ‘1’=‘1’”

Because ‘1’=‘1’ is always TRUE, the OR of the two parts of the WHERE clause is always TRUE, every record satisfies the value of the WHERE clause and so the DBMS will return all records in the database.

The trick here is that the browser application includes direct user input into the command, and the user can force the server to execute arbitrary SQL commands.

# Introduction to Statistical Database Security

Statistical databases are used mainly to produce statistics about various populations. The database may contain confidential information about individuals, which should be protected from user access. However, users are allowed to retrieve statistical information about the populations, such as averages, sums, counts, maximums, minimums, and standard deviations. We will illustrate the problem with a simple example. Various techniques are discussed, for example, in [**Adam N.R., Wortmann J.C., Security-Control Methods for Statistical Databases: A Comparative Study, - ACM Computing Surveys, Vol. 21, No. 4, December 1989, 515-556,** [**https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE552/CMPE552%20Fall2021/AdamACMCS89.pdf**](https://staff.emu.edu.tr/alexanderchefranov/Documents/CMPE552/CMPE552%20Fall2021/AdamACMCS89.pdf)]

We use Person relation with a schema

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Ssn | Income | Address | City | State | Zip | Sex | Last degree |

A **population** is a set of tuples of a relation (table) that satisfy some selection condition. Hence, each selection of the Person relation will specify a particular population of Person tuples. For example, the condition Sex=’M’ specifies the male population; the condition ((Sex=’F’) and (Last\_degree=’M.S.’ or Last\_degree=’Ph.D’)) specifies the female population that has an M.S. or Ph.D. degree as their highest degree; and the condition City=’Houston’ specifies the population that lives in Houston.

Statistical database security technique must prohibit the retrieval of individual data. This can be achieved by prohibiting queries that retrieve attribute values and by allowing only queries that involve statistical aggregate functions. Such queries are sometimes called statistical queries.

# Introduction to Statistical Database Security (Cont 2)

It is the responsibility of a database management system to ensure the confidentiality of information about individuals, while still providing useful

statistical information to users. Violation of privacy can be illustrated by the following example.

In some cases it is possible to infer the values of individual tuples from a sequence of statistical queries. This is particularly true when the conditions result in a population consisting of a small number of tuples. As an illustration, consider the following statistical queries:

Q1: SELECT COUNT(\*) FROM PERSON

WHERE <condition>

Q2: SELECT AVG(Income) FROM PERSON

WHERE <condition>

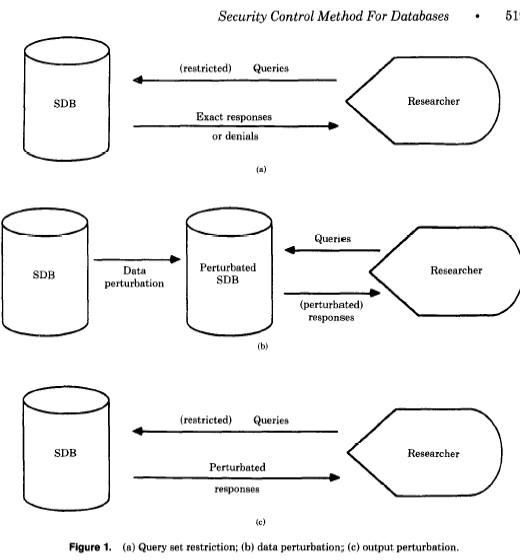
Now suppose that we are interested in finding the salary of Jane Smith, and we know that she has a Ph.D. degree and that she lives in the city of Bellaire, Texas. We issue the statistical query Q1 with the following condition:

(Last\_degree=’Ph.D.’ and Sex=’F’ and City=’Bellaire’ and State=’Texas’)

If we get a result of 1 for this query, we can issue Q2 with the same condition and find the salary of Jane Smith. Even if the result of Q1 on the preceding condition is not 1 but is a small number – say 1 or 2 – we can issue statistical queries using functions MIN, MAX, and AVG to identify the possible range of values for the salary of Jane Smith.

The possibility of inferring individual information from statistical queries is reduced if no statistical queries are permitted whenever the number of tuples in the population specified by the selection condition falls below some threshold. Another technique is to prohibit sequences of queries that refer repeatedly to the same population of tuples. It is also possible to introduce slight inaccuracies or noise into the results of statistical queries deliberately, to make it difficult to deduce individual information from the results. Some approaches to statistical database security are shown on the following figure from Adam and Wortmann:

# Introduction to Statistical Database Security (Cont 3)



# Flow Control

Flow control regulates the distribution or flow of information among accessible objects. A flow between object X and object Y occurs when a program reads values from X and writes values into Y. Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects. Thus, a user cannot get indirectly in Y what he or she cannot get directly in X. Active flow control began in the early 1970s. Most flow controls employ some concept of security class; the transfer of information from a sender to receiver is allowed only if the receiver’s security class is at least as privileged as the sender’s.

# Flow Control (Cont 1)

A flow policy specifies the channels along which information is allowed to move. The simplest flow policy specifies just two classes of information: confidential (C) and non-confidential (N), and allows all flows except those from class C to class N. This policy can solve the confinement problem that arises when a service program handles data such as customer information, some of which may be confidential. For example, an income-tax computing service might be allowed to retain a customer’s address and the bill for services rendered, but not the customer’s income or deductions.

Access mechanisms are responsible for checking user’s authorizations for resource access: only granted operations are executed. Flow controls can be enforced by an extended access control mechanism, which involves assigning a security class (usually called a clearance) to each running program. The program is allowed to read a particular memory segment only if its class is as low as that of the segment. This automatically ensures that no information transmitted by the person can move from a higher to a lower class. For example, a military program with a secret clearance can only read from objects that are unclassified and confidential and can only write into objects that are secret or top secret.

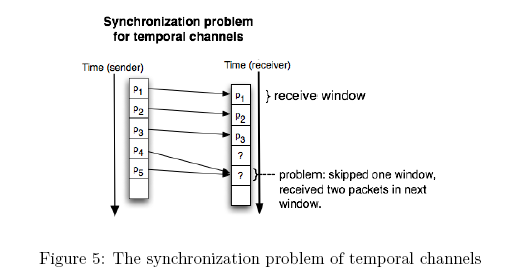
Covert channels can be classified into timing (temporal) and storage (spatial) channels. One way to think of the difference between covert timing channels and covert storage channels is that covert timing channels are essentially memoryless, whereas covert storage channels are not. With a timing channel, the information transmitted from the sender must be sensed by the receiver immediately, or it will be lost. However, an error code indicating a full disk which is exploited to create a storage channel may stay constant for an indefinite amount of time, so a receiving process is not as constrained by time.

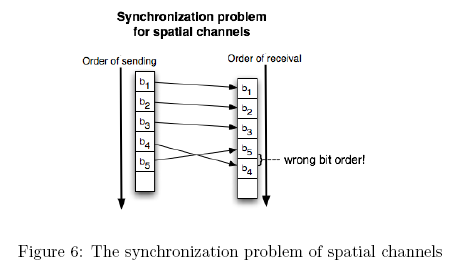
# Covert Channels in Networks

Further examples are taken from [Smeets M., Koot M., Research Report: Covert Channels. - <https://rp.os3.nl/2005-2006/p27/report.pdf> ]

In Figure 5, if a packet is received in the time window, then bit is 1, otherwise, it is 0. There may be problems related with delivery delays.

# Covert Channels in Networks (Cont 1)

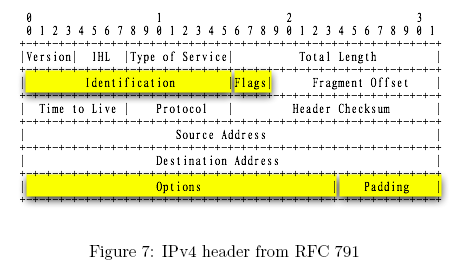




In Figure 6, packets may arrive not in the original order that results in swapping of bits.

Covert channels using IPv4 are discussed below.

# Covert Channels in Networks (Cont 2)



Format shows bit numbers.

Version = 4 (4 bits)

**Internet Header Length** (4 bits) – Length of header in 32-bit words. The minimum value is five, for a minimum header length of 20 octets (8 bit bytes), the maximum is 15.

**Type of Service** (8 bits) – Provides guidance to end system IP modules and to routers along the packet’s path, in terms of packet’s relative priority

**Total Length** – Total IP packet length, in octets

**Identification** – A sequence number that, together with the source address, destination address, and user protocol, is intended to identify a packet uniquely. Thus, the identifier should be unique for the packet’s source address, destination address, and user protocol for the time during which the packet will remain in the internet.

**Flags** – Only two of the bits are currently defined. When a packet is fragmented, the More bit indicates whether this is the last fragment in the original packet. The Don’t Fragment bit prohibits fragmentation when set.

**Fragment Offset** – Indicates where in the original packet this fragment belongs, measured in 64-bit units. This implies that fragments other than the last must contain a data field that is a multiple of 64 bits in length.

**Time to Live** – Specifies how long, in seconds, a packet is allowed to remain in the internet. Every router that processes a packet must decrease it by at least one, so the TTL is somewhat similar a hop count

**Protocol** – Indicates the higher level protocol, which is to receive the data field at the destination; thus, this field identifies the type of the next header in the packet after the IP header.

# Covert Channels in Networks (Cont 3)

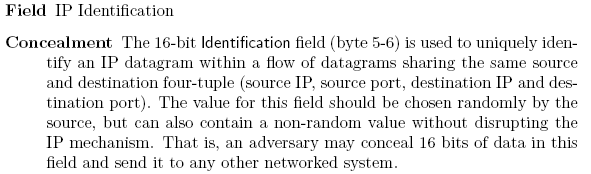
**Header Checksum** – An error detecting check code applied to the header only. Because some header fields may change during transit (e.g., time to live), this is re-verified and recomputed at each router. For purposes of computation, the checksum field is itself initialized to a value of zero.

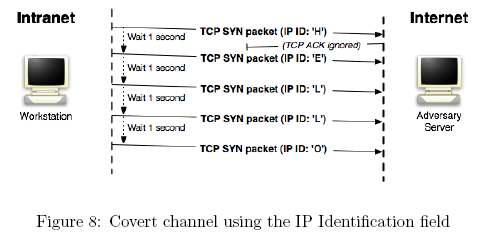
**Source Address** – Coded to allow a variable allocation of bits to specify the network and the end system attached to the specified network.

**Destination Address** – Same characteristics as source address

**Options (variable)** – Encodes the options requested by the sending user; these may include security label, source routing, record routing, and time-stamping.

**Padding (variable)** – Used to ensure that the packet header is a multiple of 32 bits in length





# Audit

We follow [**Hassan A. Afyouni, Database security and auditing, Thomson, 2006, Chapter 7**]

The auditing practice that is most often publicized is the review of an organization’s documents such as the financial statements to make sure that every change to the account is recorded and is accurate. An audit also assures that all company transactions comply with government regulations and laws. Most importantly, an audit can be conducted as a review of the enforcement of security policies and procedures. All audits take place in an auditing environment (database auditing environment), which includes:

* Objectives – An audit without a set of objectives is useless. Objectives usually are set by the organization, industry standards, or government regulations and laws
* Procedures – To conduct an audit, step-by-step instructions and tasks must be documented ahead of time.
* People – Every auditing environment must have an auditor, even in the case of automatic audit. Other people involved in the audit are employees, managers, and anyone being audited
* Audit entities – This include people, document, processes, systems, activities, or any operations that are being audited
* Database – Database activities, data changes, data accuracy, data files, and operating system

Security measures are inseparable from auditing.

# Auditing Process

The Auditing process (AP) ensures that the system is working and complies with the policies, standards, regulations, or laws set by the organization, industry, or government. It differs from Quality assurance (QA) and Performance monitoring (PM). The QA process is active during the development phase of the product and before implementation of the system. It is aimed to assure that product is working correctly. The PM process is active when the product is commissioned into production and is aimed to monitor response time. The AP is active after product is commissioned into production but is aimed on verification that the product or system complies with policies, laws, and standards.

# Auditing Process (Cont 1)

The AP has the following phases:

In system development life cycle (SDLC), - when the system is planned, analyzed, developed, tested, and implemented, - policies, laws, and industry standards must be incorporated as part of the system requirements and specifications. After the system is in production, the AP starts. The first phase is to understand the objectives of the audit and plan a procedure to execute the AP. In this phase you are identifying what should happen. The second phase is to review, verify, and validate the system according to the objectives set in the previous phase. Divergence from the auditing objectives is documented. In this phase you are observing what is happening. The last phase is to document the results of the audit and recommend changes to the system. In this phase you are identifying the changes that are needed.

# Auditing Objectives

Here are the top ten database auditing objectives:

* Data integrity
* Application users and roles – Ensure that user are assigned roles that correspond to their responsibilities and duties
* Data confidentiality – Identify who can read data and what data can be read
* Access control – Ensure that the application records time and duration when a user logs onto the database or application
* Data changes – Create an audit trail of all data changes
* Data structure changes – Ensure that the database logs all data structure changes
* Database or application availability – Record the number of occurrences and duration of application or database shutdowns (unavailability) and all startup times. Also, record all reasons for any unavailability
* Change control – Ensure that a change control mechanism is incorporated to track necessary and planned changes to the database or application
* Physical access – Record the physical access to the application or the database where the software and hardware resides
* Auditing reports – Ensure that reports are generated on demand or automatically, showing all auditable activities

# Audit Classification

Internal – is conducted by a staff member of the company being audited

External – is conducted by a party outside the company that is being audited

Automatic – is prompted and performed automatically

Manual – is performed completely by humans

Hybrid – is a combination of automatic and manual audits

# Audit Types

Financial, Security, Compliance (checks compliance with standards), Operational (verifies that an operation is working according to the policies of the company), Investigative (performed in response to an event, request, threat, or incident to verify the integrity of the system), Product (to ensure that the product complies with the industry standards), Preventive (to identify problems before they occur).

# Benefits and side effects of auditing

Benefits:

* Enforces company policies and government regulations and laws
* Lowers the incidence of security violations
* Identifies security gaps and vulnerabilities
* Provides and audit trail of activities
* Provides another means to observe and evaluate operations of the audited entity
* Provides a sense or state of security and confidence in the audited entity
* Identifies or removes doubts
* Makes the organization being audited more accountable
* Develops controls that can be used for purposes other than auditing

# Side effects

Frequent audits can cause the following:

* Performance problems due to preoccupation with the audit instead of the normal work activities
* Generation of many reports and documents that may not be easily or quickly disseminated
* Disruption to the operations of the audited entity
* Consumption of resources, and added costs from downtime
* Friction between operators and auditor
* From a database perspective, could degrade the performance of the system; might also generate a massive number of logs, reports, and data that require a system purge

# 