1998 International Computer Symposium
Workshop on Cryptology and Information Security
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

DATA SECURITY FOR ENTERPRISE APPLICATIONS – A UNIVERSAL APPROACH

Nita Sarang

CMC Limited, CMC House, C-18 Bandra-Kurla Complex, Bandra (E), Mumbai - 400 051, India
Email : nita@indiawatch.org.in

ABSTRACT

Securing application data and controlling its access is one of the key issues in implementing applications. Very often we design and even deploy systems without the security layer. Is security really an add-on functionality?

Described is a generic implementation of security that can be uniformly used and can also be easily added to existing applications. The paper covers common security issues, the need for controlling functional access through functional classes, and also describes a typical implementation of the same. The paper also addresses authorizations and routing. Adding authorization functionality to existing applications is easy and implemented with small efforts to add real value to the application. A typical implementation of authorities and associated workflow in applications is also discussed.

The paper also touches upon the issues and concerns for securing on the WEB.

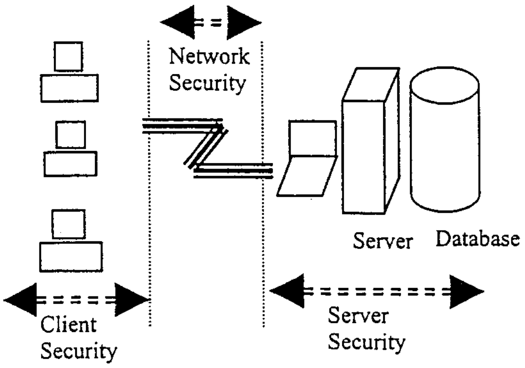
1. INTRODUCTION

As businesses are expanding, the IT requirements of the business are exponentially growing; end-users are looking towards total enterprise solutions within the infrastructure, environment and framework of their existing highly customized and isolated business applications. As the user-group for such enterprise systems grows, the need to secure data becomes more and more critical. A good number of systems currently deployed provide minimal or no data security.

Security add-ons are readily available to improve on the standard security requirements of applications. Controlling access to data based on certain roles and responsibilities is application specific, much more involved and best defined individually based on application requirements and needs. This calls for a generalized solution that can be customized as required and integrated with minimal efforts in existing applications.

2. THE SECURITY CONCEPT

A typical application or system consists of various layers; viz. client, network, server; etc. based on the deployment architecture. Clearly, security at all such application layers is necessary and important.



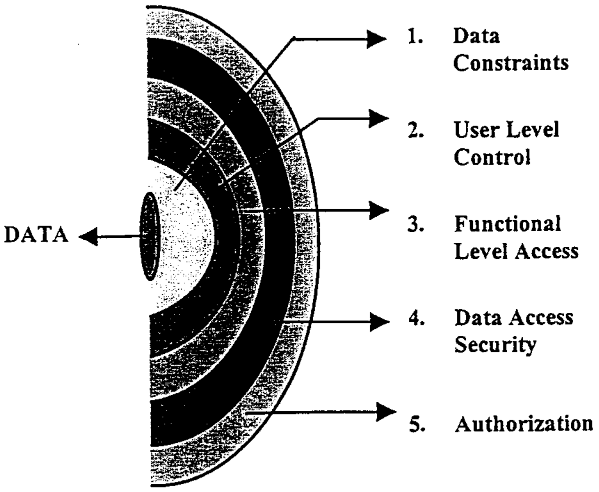
Any security system architecture must then address security issues at each of the layers of client, network and server (and WEB where applicable) to cover the complete gamut of security levels; viz.

- *System Security* for client workstation and database server
- *Database Security* through database administration
- *Application Security* through authorized users and password protection
- *Data Security* on application data

Data Security is the focus of discussions. All other levels of security are standard qualifying factors for security and achieved through system/database administration and the integration of security add-ons.

2.1 Data Security

Data Security is an application level security that controls access to application data – be it create, update, delete or view based on the users and their roles in the business application. Again, data security in applications can be implemented through various levels based on the genuine need and criticality of data. A completely secured data system falls through five levels of data security to determine access to application data.



The diagram describes the levels in the order of implementation.

Data Constraints are the basic validations on data for field/occurrence level checks and database integrity. Such constraints are foremost in specifying business rules and currently provided by all applications. In today's visual and 4GL application development environments, any model driven approach for application development makes it easy to define such basic data constraints at the application model level. At a more detailed level, the same is also specified globally through procedures and services.

User Level Control forms the basis for all other levels of security. It is the first step to validating system users and providing entry into the system. User level control is easily achieved through standard user definition screens and a system "*logon*" functionality. Encryption/ decryption of user passwords and their maintenance is effortlessly provided through the use of <ENCRYPT>, <DECRYPT> triggers, cryptology and the definition of common services to handle the same.

Other three levels of data security are the theme of discussion and will be discussed in depth in the following sections.

The generalized solution discussed is based on the above security model and can be used to selectively enable security layers as determined by the business requirements.

The criticality of implementing security as an add-on is not affected by the need to add on new definitions and configuration/administration front-ends but determined by the level of change required in existing application components to incorporate the same system-wide and into the already existing, well tested and implemented functionality. As will be seen, the proposed solution integrates with absolutely minimal changes to code and design.

3. IMPLEMENTING FUNCTION ACCESS

The business requirements of today's IT solutions are more and more turning towards work distribution and the identification of users and the association of specific roles and responsibilities at their level of operation. It is then imperative to assign each user with a specific role in the application process that determines the workspace of application data that the user is responsible for maintaining.

A specific role of a user is defined in terms of the application processes or functions and the actions or operations on the same that the user has access to. Thus, a need to associate users with functions or business processes of the application. At a more detailed level, it may also be necessary to define the action or operation list on the accessible functions available to the user. The concept of defining users along with their access permissions to the application processes/functions is called *functional security*.

Since roles and responsibilities cannot relate one-to-one with system users, a number of users would fall within the category of each role that can be defined. That is users can be classified into groups where all members of the group share the same roles and responsibilities. Defining user-wise responsibilities is not necessary. Defining user groups and the functional access permissions at the group level is sufficient. All users assigned to a specific group will maintain common system access permissions. The concept of user groups to define functional access to the system is called *functional classes*.

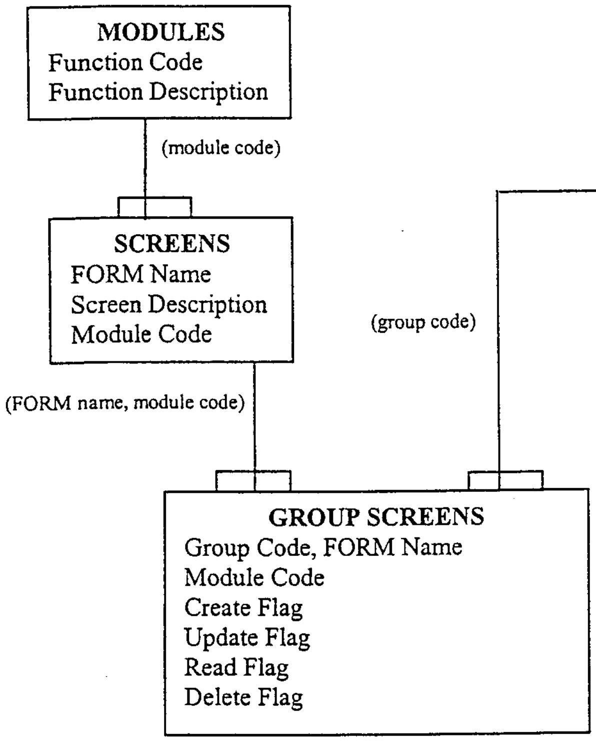
Functional classes and functional security is easily implemented in existing applications though the following steps:

1. Definition of a simplistic data model to define users, user groups and mapping between system functions.

2. Administration/configuration front-ends to specify the user groups and define the functional access permissions.

3. Simple modifications to code to incorporate the function access check.

3.1 The Data Model



USERS	User Level Control
GROUPS	User Group (Role Identification)
MODULES	Application Functions
SCREENS	Entry FORM Identifier for Function Screen
GROUP SCREENS	Group-wise Functions and Access Rights

The data model is simplistic and easily added to existing application model definitions.

3.2 Modifications to Code

Function entry points must include an access validation check to determine whether access to the function is available to the user requesting it.
The check is easily implemented through a simple access_check service that checks and sets access permissions from GROUP_SCREENS as global flags that are checked as required to determine access at the operation/action level.

Services are self-contained components of code that are centrally available for applications to use as required.

A library of service components together with the global variables (denoted by '$$') will finally form the core of the application system.

The service is:

```
$$group_code is available
service  "SYS_ACCESS".SET($formname,
         $$ACCESS_STATUS)
if ( $$ACCESS_STATUS = -1 )
         service "SYS_SERV".MSG_TEXT(
                   "M4515", $$TABLE_NAME, %\
                   $$MSG_STATUS ) ;      access
                                         denied
endif
```

Again, access checks at the individual action level can be performed through a 'check' operation on the same service.

```
service "SYS_ACCESS".CHECK
         ($$ACCESS_STATUS )
```

A pre-defined application user group may be supplied along with the application. The group must have administrative permissions to configure the functional classes and user access permissions for the user groups. This also provides the flexibility to configure the security layer to individual requirements of organizations.

4. IMPLEMENTING DATA ACCESS SECURITY

Defining roles and access permissions to functions only guarantees that application data is generated and maintained by the right people through the right functions. Functional Access security cannot be used to restrict read access based on data values stored in the system.

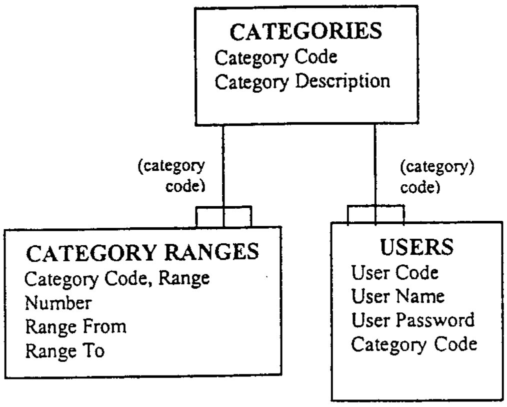
Critical data must also be protected from view access through view mode, queries and reports. Selectively controlling access to such data at the table level (business transactions) is easily achieved through the definition of categories for data access and a common criteria for ranges of data values accessible for each category. The category and its set of data values then govern the access permissions to data.

Typical categories are amount fields, departments, account heads, vendors, customers, etc.

Data access security is easily implemented in existing applications though the following steps:

1. Definition of a simplistic data model to define users, categories and category range values.
2. Administration/configuration front-ends to defining categories and its ranges.
3. Implementation through view definitions.
4. Simple "read time" modifications to code to incorporate the access check.

4.1 The Data Model



CATEGORIES	Access Category (One per User)
CATEGORY RANGES	Category Ranges for Data

Multiple category ranges can be specified. Ranges must be disjoint and overlapping values are not permitted.

The system may be provided with a pre-defined category that allows complete access to data.

4.2 Implementation Details

Read access to data based on ranges of data values is achieved through the definition of views for each category based on the ranges specified. Thus views must be defined for all transactions. The views are then used while retrieving records from the associated tables. The same strategy is used for reports so that only accessible data is displayed.

Since every user is uniquely identified by a category, an application logon session uniquely identifies the access category. This category is then used in determining all data access ranges. View names are formed as
'v'+'category code'+'transaction id'

Example: Category: Branches (code = 2)
Transaction: Purchase Order (id = PO)

The view definition for the transaction will be:

```
create view v2PO (srl_no) as
select (srl_no from po_hdr where
(branch between 100 and 200) or
(branch between 500 and 600))
```

Transaction: Journal Voucher (id = JV)

```
create view v2JV (srl_no) as
select (srl_no from jv_hdr where
(branch between 100 and 200) or
(branch between 500 and 600))
```

4.3 Modifications to Code

Modifications are limited to accessing appropriate data from the view definitions based on the logon user's category. This is implemented through a single service call in the read trigger of the outer-most (primary) transaction entity.

```
$KEYLIST$ = "location ; sls-xref_num"
itemlist /id $KEYLIST$
service "TRANS_SERV".READ ($KEYLIST$,
        $TRANSACTION_ID$, $READ_STATUS$ )
```

$$CATEG_ID is available.
$KEYLIST$ is the list of key fields.

The service itself will be coded as follows :

```
if ( $$cateq_id = 1 )
        read
else
        read while < key field > in %\
        ( select < key-field > from
                < view-name > )
endif
```

The implementation calls for minimal changes to existing code.

While the above solution achieves data access control as desired, the implementation is based on a number of assumptions and limited by the need to define one view per transaction.

Assumptions:

1. The category field used for access control maintains consistency in its name and data type across all entities in the application data model.

2. The entity and key names for transactions follow standard naming conventions.

Limitations:

1. One view definition per transaction.

2. Multiple category restrictions per user not supported.

Enhancements:
The data model described above is aimed at a first level implementation. To provide complete control for multiple categories per user, access restrictions can be based on defining selection criteria along with permitted ranges for the same. Such selection criteria are then defined at the transaction level. This permits users to have multiple access categories based on the transaction requirements. Such level of control is implemented through two tables:

CRITERIA MASTER
criteria code, criteria name, datatype, size of datatype

Each user is associated with multiple criteria; viz. customers, vendors, account heads, branch; etc.

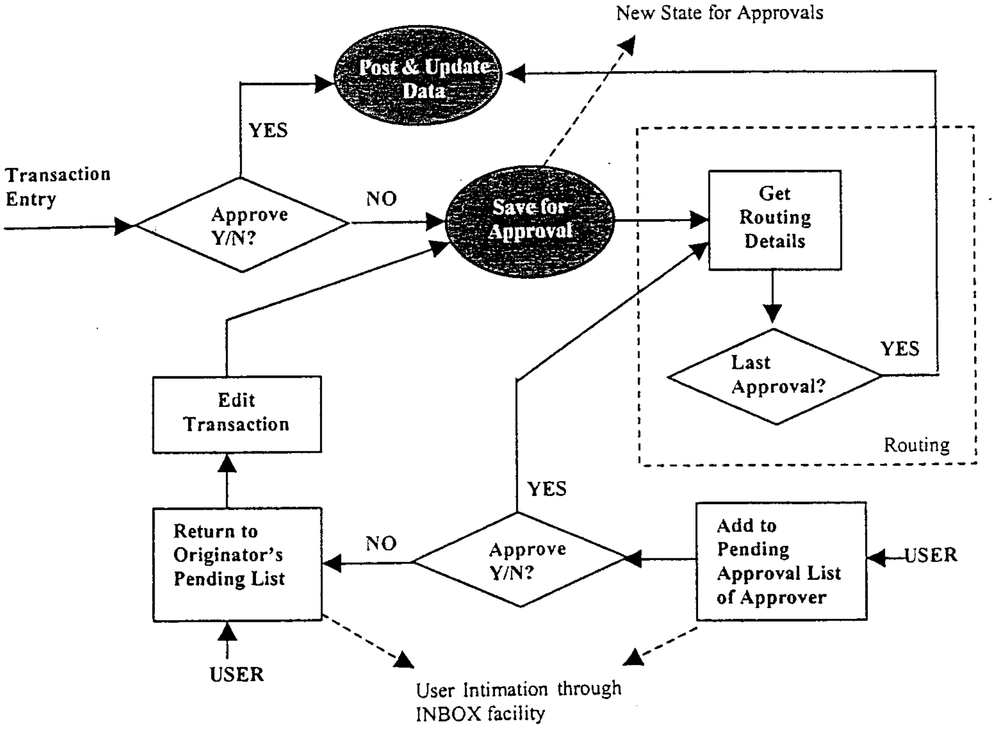
TRANSACTION-CRITERIA MASTER
transaction-id, criteria code
Based on the transaction-id, related criteria are picked up to create the view definitions.

Developers can start with a minimal implementation and then extend further to provide full versatility.

5. IMPLEMENTING AUTHORITIES

Authorities are the highest level of security implemented in applications. Authority means the ability to approve transactions entered into the system before the results of the same are effected on system data. Financial systems, in particular, necessitate various types of approvals. Levels of approval are determined by the business requirements – one approval, multiple approvals in sequence or at a much broader level, even multiple approval types in parallel; e.g. both of, functional and financial approval. This calls for a standard approval process that can handle approvals based on business requirements.

The approval process introduces its own workflow with a need to route the transaction information as defined by the approval routing. A typical approval workflow in terms of states and transitions is depicted below:



In terms of enhancements to existing applications, the above implementation of authorities relates directly to the following changes:

1. Addition of a new status to indicate 'Save for Approvals'. That is, the transaction is saved but the actions/processing and related updates; e.g. books of accounts, are not effected.
2. Deferment of "store" time updates until after approval.
3. A routing service.
4. A user notification through an "INBOX" functionality.

5.1 Modifications to Code

For applications that follow a component-based design, "save" time changes for approvals are easily incorporated and involve, first and foremost, the addition of a new transaction status in the system.

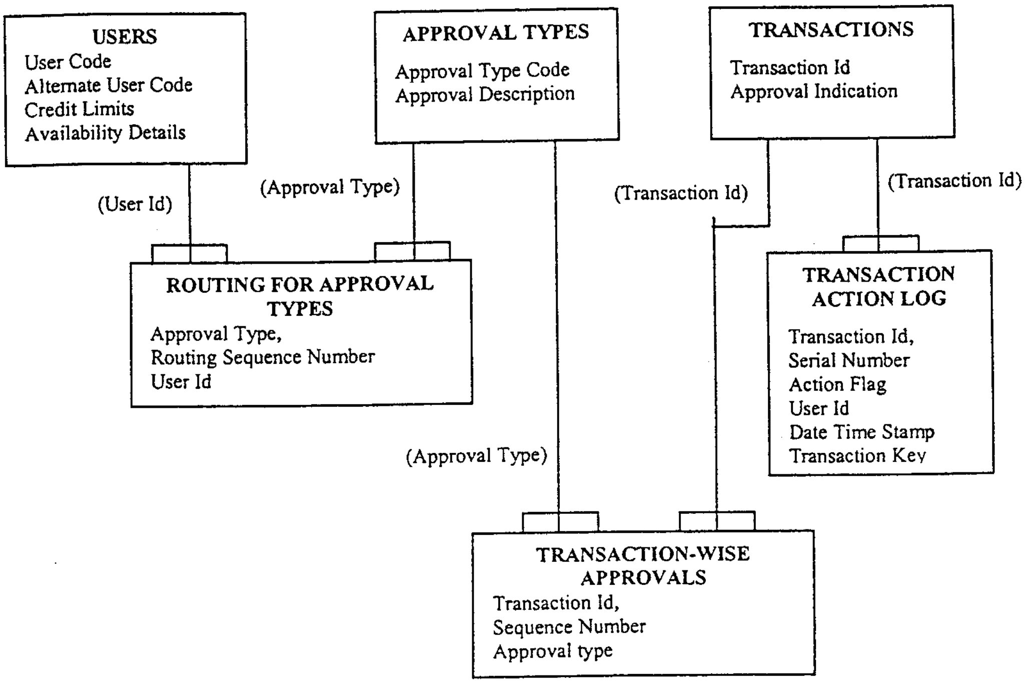
The system then provides for the following states for transactions:

• saved but not posted • posted	Already provided in existing system implementations
• saved for approvals	New for approvals

All processing involved with posting transactions is deferred until the approval process is complete. The business processing is then easily shifted to a later point in the application processing logic.

Approvals per se are implemented as a new functional component that uses services to determine routing for the individual transactions. Routing and the implementation of approval types is based on a simplistic, yet functionally complete, data model that provides full flexibility in configuring approval types and approval levels for routing.

5.2 Approvals & Routing – Data Model



USERS	User control and associated details
APPROVAL TYPES	Approval categories
TRANSACTIONS	Transaction classification for Approvals
TRANSACTION ACTION LOG	Status log of transactions during approval
ROUTING FOR APPROVAL TYPES	Routing information for approval category
TRANSACTION-WISE APPROVALS	Approval Information for transactions

The workflow handles the routing from one user to the other and maintains a track of the status log for transactions under approval. The entire processing for the approval process is handled through the implementation of a generic component and is independent of the core application.

5.3 INBOX Implementation

The approval functionality is not complete without the INBOX. The INBOX is a facility to provide users with information on activities in the system that are pending action by the user. Only user specific information is displayed.

The INBOX is generically developed to support all transactions and actions on the same and serves in some way as a diary utility that enables users to determine work that is awaiting further action. The INBOX has specific use in event-driven workflow systems whereby tasks are created for users based on actions initiated by others.

The capability of today's development tools to support plug-n-play components and loosely coupled interfaces make the generic implementation of the INBOX possible. The INBOX can then display identifying data for all transactions in a common front-end. Pending action can be initiated through the same user interface. Additional details on the transactions are easily provided through drill down on individual items into the main transaction screen in "view only" mode, thereby giving full flexibility and control through a common user interface.

Security can be further enhanced to authenticate users through the use of digital signatures. Digital signatures for users are stored into the system and the same are subsequently used to authenticate the approver during the approval process.

4. WEB SECURITY

Organizations are seriously looking towards providing WEB-based solutions. End-users too are eager to exploit the World Wide Web for core applications. Any native applications that organizations build today must necessarily be WEB-enabled. Securing such applications is then a challenge. No level of application security can replace security issues on the WEB.

Tremendous research has gone into WEB security for electronic commerce and the securing strategies are more or less stabilized. Most Web browsers support some kind of information encryption mechanisms for securing the communication. The Internet connections are protected using Secure Socket Layers (SSL). Firewalls are standard.

The SSL works with "secure" connections using symmetric encryption in combination with a public key. Users must then procure digital authentication certificates to gain access to the applications after the proper exchange of key information.

Standard WebEnabler support for development enables applications to be configured to request such digital certificates from the clients. The WEB server verifies the certificate and the application is started. No specialized user-level coding to incorporate digital certificates is required.

No specific authentication by the application is required and the client browser and WEB server together handle security.

5. CONCLUSIONS

The generic implementation of application security discussed can enhance any application to provide full security for application .data. The ability to selectively apply various levels of security allows applications to customize the implementation to suit the business requirements. The data model for the various layers is simplistic and independent of the core application. Current technologies makes it even easier to implement - with the component-driven paradigm and the ability to incorporate standard functionality through services and operations. The ability to support "plug-n-play" components demands minimal interface requirements between the calling and called components.

6. REFERENCES

[1] Ronald G Ross, The Business Rule Book - Classifying, Defining and Modeling Rules; Data Base Newsletter, 1997

[2] Ivar Jacobson etal; The Object Advantage : Business Process Re-engineering with Object Technology, Addison - Wesley, 1995

[3] Ivar Jacobson etal; Software Re-use : Architecture Process and Organisation for Business Success, 1997