20

A Methodology for the development of secure Application Systems

H.A.S. Booysen[a] and J.H.P. Eloff[a]

[a] Department of Computer Science, Rand Afrikaans University, P.O. Box 524, AucklandPark, 2006 Johannesburg, South Africa

ABSTRACT

Usually if an application system requires some security features, these are either not available at all, or are incorporated by means of other software products. The security features offered by these security products are usually very limited, in the sense that the application system designer has to rely on the underlying platform's security measures, or add security features needed by the application system. This means that security is not considered until the operational requirements have been defined and the system is well into the implementation stage. This approach towards application system development poses a problem, since it is seldom possible to provide a good level of security on a retrofit basis, or in parallel but separate from the functional design. To overcome this problem, the security aspects associated with the development of an application system should be considered during the definition of user requirements and incorporated into the system during the design stages. This paper presents a methodology that addresses security requirements as part of system development, while simultaneously considering other functional requirements.

Keywords: Application system, Information security, Spiral model, CASE tools.

1. INTRODUCTION

When studying the process of analysing and designing an application system, it is evident that security in application systems is usually implemented in a form of so-called security services [1], which are developed without considering the security requirements needed by the application system. Security requirements are usually added to the system after development, because it is believed [2] that the final application system will suffer

- loss of performance with the addition of security features,
- loss of flexibility owing to restrictions and confinements in the target system's behaviour; and
- higher costs to account for analysis of the security requirements, design and implementation of the security specifications, and maintenance of security in the application system.

J. H. P. Eloff et al. (eds.), *Information Security — the Next Decade*

Furthermore, imposing security requirements on systems already in operation shortens the lifespan of the system, complicates the modification process, increases maintenance costs, raises operating overheads and diminishes the return on investment in the system.

Currently, the state-of-the-art in application system security is such, that if a particular application system requires some security services, these are either not available at all, or incorporated by means of hardware/software into the operating system or network services. The security features offered by these services may also be implemented as add-on features through some commercially available security products. The scope of these products is usually very limited [3,4], in the sense that the application system designer either has to rely on the underlying platform's security measures (which are often lacking) or has to add the security features needed by the application system.

Current system development methodologies do not consider security as part of application system development. Furthermore, security is not considered until the operational requirements have been defined and the system is well into the implementation stage [5]. This presents a problem since it is seldom possible to provide a good level of security on a retrofit basis or in parallel but separately from functional design. Even if this can be achieved, the costs involved [4] will be quite high when compared with those incurred when the security requirements are considered at the very beginning of the system development process.

The security requirements associated with the development of an application system should therefore be considered during the definition of user requirements and incorporated into the system during the design stages. A new development methodology is needed to provide an approach towards system development that addresses security requirements as part of system development, while considering other functional requirements. It is the aim of this paper to describe such a systematic development methodology towards secure application system development.

The paper is structured as follows: In Section 2 a high-level view of the new development methodology is given. Security requirements that should form part of the development methodology are described in detail in Section 3. A practical example illustrating the working of the security activities is presented in Section 4, after which concluding remarks are given in Section 5.

2. HIGH-LEVEL VIEW OF ASSDM: **A**utomated **S**ecure **S**ystems **D**evelopment **M**ethodology

A key notion underlying the creation of a security development methodology is to include security activities as part of system development. The proposed methodology therefore should include security and conventional development activities (functional requirements).

The original spiral methodology developed by BW Boehm [6,7] serves as the basis for ASSDM. The advantage of using the approach of spirals in presenting a development methodology, is that one can return to previous spirals during any stage of the development process. Each spiral contains development activities (as defined by Boehm) that can be associated with a spiral. The Risk Analysis activity is used to show the beginning of a new spiral. The original spiral model developed by Boehm is diagrammatically summarised in Figure 1.

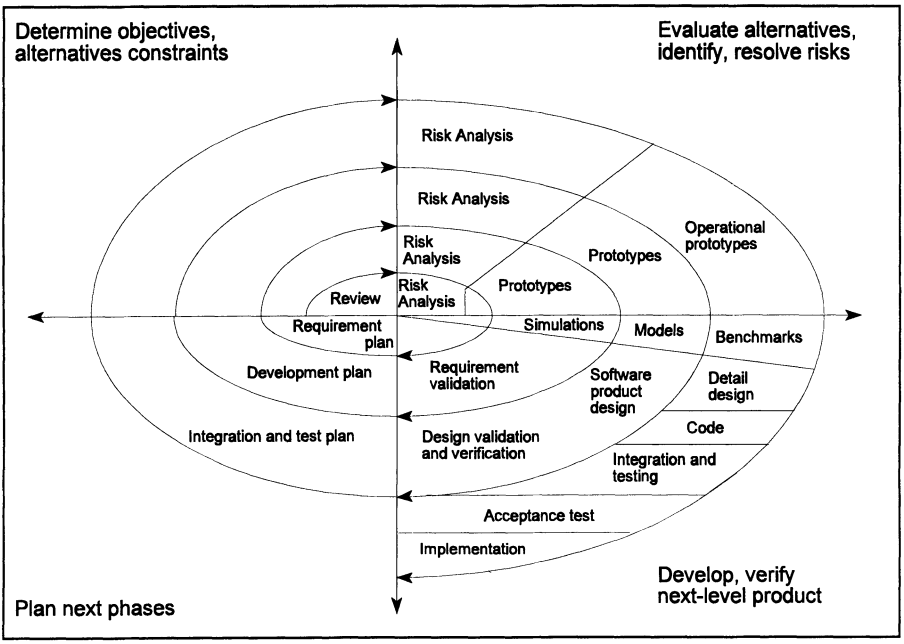

Figure 1. Original spiral model [6,7]

2.1. Identification of security activities

To include security as part of system development, it is necessary to first identify security activities that will contribute to ensure that the integrity, confidentiality and availability of the system are maintained under all circumstances. These security activities will be identified, when examining the type of data that the system will process and the functions to be performed by the system. Integrity, confidentiality and availability requirements share the objective for describing the access to objects and between objects in an application system. Access Control Mechanisms must therefore be regarded as the heart of the information security requirements of any computing system. Access Control Mechanisms have practical value in ensuring that only authorized objects and users can gain access to the objects in a system. It is also important to ensure that the objects protected by the mechanism are classified correctly.

The main problem when developing secure application systems is not only to introduce access controls, but to correctly assess sensitive information contained in the application, with regard to the confidentiality of that information. As information is added continuously to various databases of an application system, the only way to effectively assess all the information contained in an application system, is to study how the information flows within the application system. Various dataflow diagrams can be used to depict the flow of information between objects in an application system. The flow of information in an application system is summarised by means of an object matrix. In an object matrix objects from which information flows are mapped onto objects to which information flows. Valid information flows between objects are summarised by means of a revised object matrix. (This is discussed in more detail in paragraph 3)

System development does not occur in isolation. It is therefore necessary to involve concerned users, system developers and information security specialists who understand how to incorporate the identified security controls into systems while the system is still in the process of development. Additional security requirements needed by the system are identified when the appropriateness of existing security requirements are validated by means of prototypes. To ensure that the goal state defined for the system complies with policies, standards and regulations used in an organization, the system must be audited. Testing the system will ensure that the security requirements defined for the system are adhered to. A security report will contain findings of the audit and security tests.

A risk analysis is usually seen as extremely important in analysing the security features needed by a system. Unpredictable guessing at what the impact of a threat on an object may be, enhances the fact that a risk analysis is only a very small portion of what is needed to improve the total security of the application system under development.

Using the original spiral model as basis for the proposed development methodology, the security activities identified above can be placed on a security spiral. The sequence of security activities within the security spiral is explained as follows:

The security spiral is divided into four quadrants, namely:

- A "*Determine objectives, alternatives, constraints*" quadrant that helps to define the security requirements needed by the system. This is done when the sensitivity level of the application system is determined.

- An "*Evaluate alternatives, identify, resolve risks*" quadrant to help in selecting the best security development strategy, while considering security risks associated with requirements. The security prototype and Revised object matrix activities assist in selecting the best development solution.

- A "*Develop, verify next-level product*" quadrant that describes the phases involved in system development. The security requirements needed to reach the goal state are defined, for example security requirement validation, security models, information flow analysis and audit.

- A "*Plan next phases*" quadrant that combines the deliverables from the other quadrants to assist the project team and developer in planning the next spiral, by means of revising existing security requirements and identifying additional security controls needed by the application system.

This newly formed spiral is depicted in Figure 2.

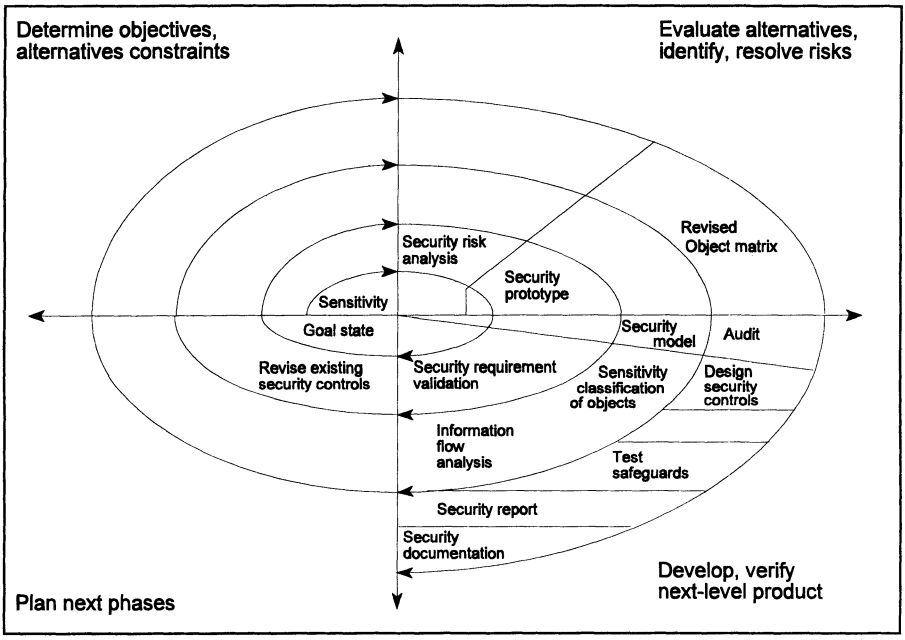

Figure 2. Security spiral

When combining the original spiral (Figure 1) and the security spiral (Figure 2) a new spiral is formed. When using the newly formed spiral to develop a system, the development and security activities contained in the original and the security spiral respectively, are considered simultaneously during system development. In this way security becomes part of system development.

3. SECURITY SPIRAL

A typical example of the security spiral can be described as follows:

To incorporate security activities into the development process of an application system it is necessary to **determine the sensitivity level of the application system** (indicated as Sensitivity in Figure 2). The degree of sensitivity of an application system depends upon the data it will process, and/or the type of functions that the software will accomplish [9]. Data sensitivity can be caused by several factors that include the value of data, the source of data and declarations about data made by the owner of the data [10,11]. The objective of this activity is therefore to determine the impact of unauthorised access to the application system.

When the sensitivity of the application system has been determined, it is necessary to **define the goal state of the application system** (indicated as Goal state in Figure 2). The goal state of any application system can be defined as the breach between the current state and the expected state of the application system. When defining the goal state of an application system, it is necessary to define it in terms of three security objectives, namely integrity, confidentiality and availability [8]. This will ensure that information in the application system is only available to, and continuously accessible by authorised parties of the system.

To improve the overall security of an application system, it is necessary to consider the risks associated with the planned security features. The planned security features were identified when the goal state of the application system was defined. **Conducting a security risk analysis** (indicated as Security risk analysis in Figure 2) will help to determine whether it would be possible to reach the goal state defined for the application system when using countermeasures (for example access control lists, security policies) currently available in the organization. A security risk analysis will also identify the security vulnerabilities and shortcomings of the application system. For example, when conducting a risk analysis it will become evident that a specific information security policy might be needed to ensure that the security of the system is maintained. A security risk analysis will also help to identify countermeasures that will ensure that the integrity, confidentiality and availability of information are maintained under all circumstances.

To ensure that the system under development performs according to its specifications, it is necessary to involve the user. The use of **prototypes** (indicated as Security prototype in Figure 2) helps to give an overview of the co-ordination of application characteristics (i.e., the user requirements) in conjunction with the underlying security features of the system. The prototype will also highlight security related problems in the design of the application system, when **security requirements** are **validated** (indicated as Security requirement validation in Figure 2).

When validating security requirements it will become evident whether additional security controls are needed, or whether existing controls need to be revised to meet the security requirements of the users. This is indicated as Revise security controls in Figure 2. An object classification scheme, for example, can be used to classify objects according to their sensitivity. In a commercial application system, objects can be classified, for example, as top secret, secret, confidential or unclassified.

A **Security model** (indicated as Security model in Figure 2) gives an overall view of the application system on a high level. Various diagrams (dataflow, context, entity relationship) are used to represent the application system visually. A dataflow diagram for example, consists of four basic objects, i.e., processes, entities, data stores and flows. A powerful feature of a dataflow diagram is that it provides the opportunity for modelling the security requirements of system functions by extending them to include **object classifications** and object interactions (indicated as Sensitivity classification of objects in Figure 2). The security class of an object consists of a weight (value) allocated by the Information Security Officer (ISO) and user, based on his assessment of the sensitivity level of the information contained in an object. The interdependence that exists between objects can also contribute to determine the sensitivity level of an object. As a commercial application contains several objects, it would be feasible to automate the process of classifying objects as far as possible [13].

Objects are connected on a dataflow diagram by means of an arrow symbol. The action that one object performs on another object is defined by the arrow symbol on the dataflow diagram. In a commercial application system the interactions between objects usually include read, write, append, update and delete actions. The arrow symbol also portrays the direction of information flow on a dataflow diagram. These actions can be used to construct an **object matrix**. An object matrix is a rectangular array in which objects from which information flows (origins) are mapped onto objects to which information flows (destinations). The entry for a particular row and column reflects the information flow action between the corresponding objects. Rows in the object matrix indicate origins of information flow, and columns indicate destinations of information flow. An object matrix shows only direct (binary) information flow between objects, in other words, information flowing between objects linked directly to one another. For example, an object matrix contains both valid and invalid binary information flows between objects. If a top secret object reads information contained in a confidential object, the binary flow between the two objects would be valid, but if a confidential object reads information from a top secret object, the binary flow would be invalid. (According to the access rules as defined in the Bell-Lapadula model [8].)

To comprehensively address all combinations of information flow in a system, it is necessary to **conduct an information flow analysis** (indicated as Information flow analysis in Figure 2) on an information flow diagram such as a dataflow

diagram. An information flow analysis also assists in considering the flow type that exists between objects not linked directly to each other, but rather indirectly by means of intermediate objects. We refer to this kind of information flow as *Compound Information Flow*. A compound information flow between objects is determined using the rationale of the "grant" right in the Take-Grant model [12]. The objective is to determine the "combined" flow type that could exist between:

> $Object_A$ and $Object_C$, $Object_A$ and $Object_D$ and $Object_B$ and $Object_D$ in the example presented in figure 3.

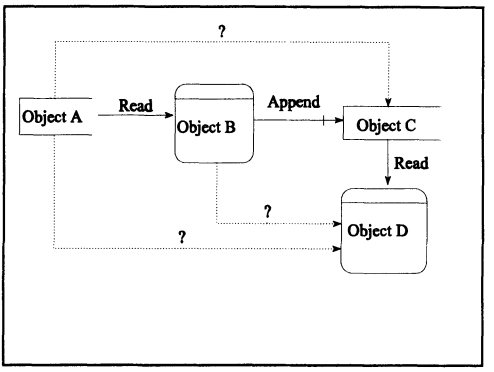

Figure 3. Example compound dataflow diagram

In determining the compound flow type that exist between $Object_A$ and $Object_C$, it is necessary to substitute the append flow type between $Object_B$ and $Object_C$ with "write". This allows one to indicate a compound flow type in terms of the actual action that occurs. As an update flow requires information to be read before it is written to another object, the update flow type is replaced by "read-write". A compound flow type is determined between the first object and the third object. The "newly" formed information flow type is then used as the "first" flow type in determining the compound flow type between the next two objects. For example, the compound information flow between $Object_A$ and $Object_D$ is determined as follows:

The information flow type between $Object_A$ and $Object_B$ is "read", and the flow type between $Object_B$ and $Object_C$ is "write" (append), we obtain a "read-write" flow type, indicating an update. Therefore the compound information flow between $Object_A$ and $Object_C$ is "Update". This "newly" formed flow type now serves as the first information flow type in determining the compound flow between $Object_A$ and $Object_D$. As the flow type between $Object_C$ and $Object_D$ is read, the compound flow between $Object_A$ and $Object_D$ would be "Read". (The combination of Update - between $Object_A$ and $Object_C$, and Read between $Object_C$ and $Object_D$.) Possible combinations of compound flow types are depicted in Table 1.

Table 1
Compound information flow type combinations

Between $Object_1$ and $Object_2$	Between $Object_2$ and $Object_3$	$\therefore$ Between $Object_1$ and $Object_3$
Read	Append	Update
Read	Update	Update
Read	Read	Read
Append	Read	Read
Append	Append	Append
Append	Update	Update
Update	Read	Read
Update	Update	Update
Update	Append	Read

The security activities performed so far have assisted in determining both valid and invalid binary and compound information flows. From a security point of view, the question arises as to when the binary and compound information flows would be valid or invalid. Valid binary and valid compound information flows are determined by using the security classes assigned to objects (indicated as Sensitivity classification of objects on Figure 2), and by applying access rules (for example Bell-LaPadula [8]) which state when a flow is valid or invalid. All valid combinations of information flow are summarized in a **revised object matrix** (indicated as Revised Object matrix in Figure 2).

In comparing the object matrix and revised object matrix, invalid information flows are identified. When these flows are examined, and the necessary actions are taken (for example, changing the security class of an object), the security state of the application system is improved.

To ensure that the system complies with standards, policies and regulations within an organization, it is necessary to **perform an audit** (indicated as Audit in Figure 2), on the system. After conducting the audit, additional security controls can be designed (indicated as **Design security controls** in Figure 2).

To show that an application system can be trusted, and that specifications are adhered to, it is necessary to perform certain **security testing** and evaluation activities (indicated as Test safeguards in Figure 2). The testing of security safeguards must focus on assuring that the user uses safeguards correctly and that

safeguards cannot be bypassed.

A **security report** (indicated as Security report in Figure 2) is used to document all test results and findings. Deficiencies in safeguards taken to improve security must be identified. Actions that are necessary to improve these deficiencies must be identified, carried out and tested.

System **documentation** (indicated as Security documentation in Figure 2) is created during any software development process. This is especially necessary for security relevant code. It is advisable to keep the documentation of the relevant security relevant code in a separate development document, owing to the sensitivity of the data contained in the documentation.

4. PRACTICAL EXAMPLE

To illustrate the practical working of the security spiral, an example is presented below. The following activities of the security spiral are illustrated:

4.1. Determine the sensitivity level of the application system and define the goal state of the application system
4.2. Perform a security risk analysis
4.3. Security prototype, security requirement validation and revise existing security controls
4.4. Security model and sensitivity classification of objects
4.5. Information flow analysis and Revised object matrix

The remainder of security activities contained on the security spiral, i.e., Audit, Design security controls, Test safeguards, Security report and Security documentation will not be discussed here, as these actions are merely an extension of the corresponding activities contained in Figure 1.

Consider the following user requirements:

> *An application is needed with a process that can calculate salaries for the employees of a large company. There is an existing database containing employee data, for instance personal data and rate per hour paid. The process appends salary data to a data file. A salary clerk needs access to the salary data so as to resolve ad-hoc enquiries, for example average salaries.*

4.1. Determine the sensitivity level of the application system and define the goal state of the application system

Using the security spiral, and having studied the user requirements, the developer decides that the application should be regarded as sensitive, as the impact of unauthorized access to the application system can result in the

disclosure of confidential data (salary data) to unauthorized parties. Currently, a manual system is used to satisfy user requirements. The goal of the new application would be to ensure that information is only available to, and continuously accessible by authorised parties of the system. The new application system will also ensure greater efficiency in the personnel department while conducting their day-to-day business.

4.2. **Perform a security risk analysis**

Security risks that can be associated with the application system are as follows:

- Accidental and/or deliberate modifications to employee and/or salary data;
- Disclosure of salary data to unauthorized employees and/or outsiders;
- Incorrectly classified system objects (process, data file, external entity), causing invalid information flows;
- Lack of security controls to detect the presence of a Trojan horse; and
- Inadequate testing of security controls.

4.3. **Security prototype, security requirement validation and revise existing security controls**

In building a prototype of the system, security requirement validation has indicated that the following additional security controls are needed to reduce the risks associated with the system to an acceptable level:

- An access control mechanism which provides protection against invalid binary and invalid compound information flow. In considering information flows while developing the system, risks of accidental and/or deliberate modifications to data, and disclosure of data to unauthorised employees and/or outsiders, will be reduced.

- An object classification scheme which takes the interdependence between objects into consideration, will assist in reducing the risk of classifying objects incorrectly. If objects are classified incorrectly, protection provided by the access control mechanism, will not ensure that only authorized objects can gain access to objects.

- An automated tool, eg. a CASE tool, that will automate mundane activities, in order to limit human errors in time-consuming activities, such as testing of security controls.

4.4. **Security model and sensitivity classification of objects**

Using a system development tool, for example a CASE tool, the designer transforms the user requirements into a visual representation as depicted in Figure 4.

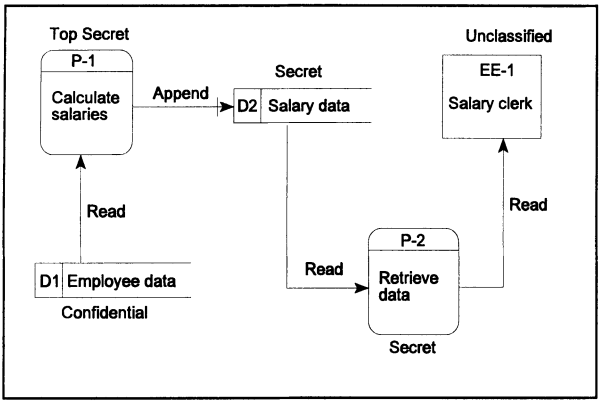

Figure 4. Example dataflow diagram

In classifying objects contained on the dataflow diagram according to their sensitivity and interdependence on another, the ISO in conjunction with the system developer and users, have indicated that the following security classes should be assigned to the process, data files and external entity objects contained in Figure 4:

Calculate salaries (process): Top Secret
Retrieve data (process): Secret
Employee data (data file): Confidential
Salary data (data file): Secret
Salary clerk (external entity): Confidential

4.5. **Information flow analysis and Revised object matrix**

In conducting an information flow analysis as described in the previous section, the following object matrix and revised object matrices are constructed. Recall that an object matrix contains both valid and invalid binary and compound information flows. A revised object matrix contains only valid binary and valid compound information flows.

Object Matrix					
	Calculate salaries	Employee data	Retrieve data	Salary data	Salary clerk
Calculate salaries			Read	Append	Read
Employee data	Read		Read	Update	Read
Retrieve data					Read
Salary data			Read		Read
Salary clerk					

Revised Object Matrix					
	Calculate salaries	Employee data	Retrieve data	Salary data	Salary clerk
Calculate salaries					
Employee data	Read		Read	Update	Read
Retrieve data					
Salary data					
Salary clerk					

When the designer compares the Object Matrix and the Revised Object Matrix, the following invalid binary information flows can be identified:

- ✎ Information flow between "Retrieve data" and "Salary clerk"; and
- ✎ Information flow between "Calculate salaries" and "Salary data".

The information flow between "Retrieve data" and "Salary clerk" is invalid, as the Salary clerk can only read information which has a confidential or unclassified clearance. To make the flow valid, the security class of the Salary clerk needs to be raised to be at least the same as the security class of the "Retrieve data" object, i.e., Secret.

The information flow between "Calculate salaries" and "Salary data" is invalid as information is flowing from a Top secret object to a Secret object. This can potentially be solved by inserting a sanitizer object between the "Calculate salaries" and "Salary data" objects. A sanitizer object will ensure that only information with the same security class as that of the receiving object are filtered through to the receiving object. Therefore, inserting a sanitizer object (SP-1) between the "Calculate salaries" and "Salary data" objects will ensure that only secret information is allowed to flow to the "Salary data" object.

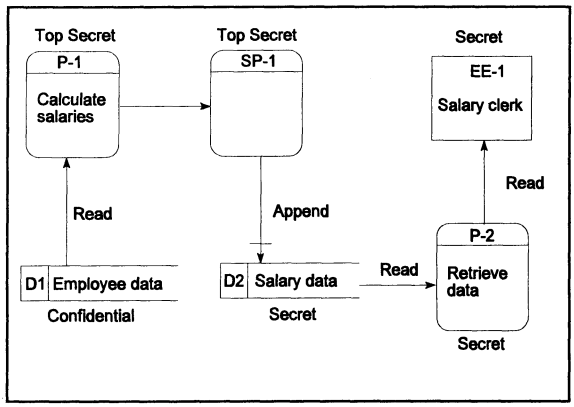

Figure 5. Revised dataflow diagram

When the designer has made the changes as described above, invalid

re-analysed, to ensure that identified risks have been addressed. The revised dataflow diagram is depicted in Figure 5.

The Revised object matrix will assist in ensuring that only authorised users can gain access to system objects. This will assist in minimising the risks of accidental/deliberate modification to data, and disclosure of data to unauthorised personnel. The insertion of a sanitizer object, assists in reducing the risk of inadequate security controls. Thereby improving the security state of the application system.

5. CONCLUSION

This paper showed how the existing spiral systems development life cycle can be enhanced to include security concepts and tools. In combining security activities with traditional development activities, as proposed by the security spiral, security aspects are considered during the definition of user requirements and incorporated into the system during the design stages. This allows security to become part of system development.

The advantage of using a CASE tool when developing a system has several benefits to the security state of the application system under development. Firstly, it potentially assists in assigning a security class to an object while developing the system. Secondly, the information flow activity can be conducted automatically, thereby eliminating human errors.

REFERENCES

1. Muftic S, Hatunic E, "CISS: Generalized Security Libraries", Computers & Security (11), 1992.
2. Baskerville R, "Designing Information Systems Security", Addison-Wesley Publications, 1983.
3. Baskerville R, "Information Systems Security Design Methods: Implications for Information Systems Development", ACM Computing Surveys (25) 4, December 1993.
4. Cresson Wood C, "Principles of Secure Information systems design with groupware examples", Computers & Security (12), 1993.
5. Ettinger JE, "Information Security", Chapman & Hall, 1993.
6. Boehm BW, "A Spiral model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes (11) 4, 1986.
7. Boehm BW, "Applying process programming to the Spiral model", Proceedings of the 4th International Software Process Workshop, 1988.
8. Pfleeger CP, "Security in Computing", Prentice-Hall International, 1988.
9. Tompkins FG, Rice R, "Integrating Security Activities into the Software Development Life Cycle and the Software Quality Assurance Process", Computers & Security 5 (5), 1986.

Computers & Security 5 (5), 1986.
10. Denning DE, "Cryptography and Data Security", Addison-Wesley Publishing Company, 1983.
11. Denning DE, "A Lattice Model of Secure Information Flow", Communications of the ACM, 1976.
12. Lipton RJ, Snyder L, "A Linear Time Algorithm for Deciding Subject Security", Journal of the Association for Computing Machinery, 24 (3), 1977.
13. Booysen HAS, Eloff JHP, "Classification of objects for improved access control", Submitted for publication in Computers & Security.