Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.016120
Article

Modernization Framework to Enhance the Security of Legacy Information Systems

Musawwer Khan[1], Islam Ali[1], Wasif Nisar[1], Muhammad Qaiser Saleem[2], Ali S. Ahmed[2], Haysam E. Elamin[3], Waqar Mehmood[4] and Muhammad Shafiq[5,*]

[1]Department of Computer Science, COMSATS University Islamabad, Wah Campus, Pakistan
[2]College of Computer Science and Information Technology, Al Baha University, Al Baha, Saudi Arabia
[3]Department Information Technology, Faculty of Computer Science and Information Technology, University of Jeddah, Jeddah, Saudi Arabia
[4]Department of IT and Computer Science, PAF-Institute of Applied Sciences and Technology, Haripur, Pakistan
[5]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, 38541, Korea
*Corresponding Author: Muhammad Shafiq. Email: shafiq@ynu.ac.kr
Received: 23 December 2020; Accepted: 14 April 2021

Abstract: Due to various issues such as lack of agility, low performance, security issues, and high maintenance costs, the organization replaces its legacy information system (LIS). However, with the expansion of information technology, the security of the old system has received great attention. The protection of legacy data and information is critical to the organization. However, achieving safety through modernization, redevelopment, or redesign of LIS is a time-consuming and costly solution, especially in small and medium enterprises (SMEs). In addition, newly developed systems often lose inherent business rules, data integrity, and user trust. In this paper, we propose a Security Modernization Framework (SMF) to inject security measures into LIS without modernizing the existing solution. Fundamentally speaking, SMF is a collection of methods and technologies that enhance the security structure of LIS to protect applications and old data. SMF consists of two layers of security control: data audit trail and user access authorization. This contribution has two key advantages. First, it can help SMEs protect their data and applications from unauthorized access. Second, it is the lowest cost solution to implement security measures in LIS instead of modernizing or replacing the system. SMF has used the oracle technology for authentication, but the examples are also shown in pseudocode to facilitate easy positioning of other technologies.

Keywords: Modernization; legacy information system; security; SMEs

1 Introduction

The traditional information system (LIS) is a data-intensive system, composed of a large number of data files, and has significant resistance to alteration [1]. LIS resists changes caused by changing business needs [2,3]. The traditional system can meet the business needs of the enterprise and meet the functional

requirements, but it lacks architecture standards [4]. Although LIS lacks various quality attributes such as availability, maintainability, and interoperability, security is one of the key attributes it lacks. Over time, access vulnerabilities will appear in LIS, because security is one of the most targeted areas of unauthorized agents. Intruders use these vulnerabilities to access valuable data. Software security means that although software failures continue to occur in the real world, the software should still run correctly in the case of harmful attacks [5]. Organizations can minimize information risks by implementing an effective information system security infrastructure [6]. Today's small and medium enterprises urgently need security systems, even if many current systems lack security attributes. As we all know, SMEs are the most important and critical part of the world economy [7]. The security and protection of information systems have become very important for enterprises, especially small and medium-sized enterprises. In small and medium-sized enterprises, security is considered to be one of the main factors for economic success [8]. The security of LIS can be improved by modernizing the entire legacy system, but this is a difficult task. Many transformation and modernization models have been proposed to meet this challenge. However, replacing the old system will produce a technically inefficient and difficult to manage system [9]. Due to a lot of redevelopment work, it is not suitable to use various technologies to solve the vulnerabilities in legacy applications [10]. Therefore, switching from the old business process to the new business process is indeed a tedious task [11]. Replacing the system also risks losing a lot of business knowledge [12]. Therefore, due to these challenges, many researchers have proposed various solutions to improve the quality of existing LIS, rather than undertake the arduous task of replacing these systems. In this article, we propose a model that integrates security measures without the need to modernize or redesign the existing LIS. This model is a collection of security control methods and technologies that can help organizations overcome security vulnerabilities and control threats to corporate assets. The structure of this article is as follows. Related work has been discussed in Section 2. Section 3 introduces the proposed framework, namely SMF and its implementation details. The verification process, case studies, and results have been explained in Section 4. Finally, Section 6 summarizes this article.

2 Related Work

In this section, the security and access control models suggested by various researchers are discussed. Data and security are regarded as major risks, especially those that contain critical data [13]. The author proposed a Dynamic Safety Adaptive Controller (DSAC) in [14]. The controller dynamically adjusts static safety rules to configure and adjust existing conditions related to legacy systems. The author uses a whole to determine and specify access rules for any emergencies. In [15], another public/private key-based method was proposed. According to the author, when a user wants to access the old system, the attestation will be sent to the application server, which uses its private key to decrypt the data. Then, the retrieved authentication data will be further passed to the target traditional application, which will ultimately authenticate the user. Xiaowei Li et al. [10] discussed the identification of security vulnerabilities in traditional web applications. The author explains two types of security testing techniques, namely static analysis and runtime protection. In static analysis, vulnerabilities related to the source code will be identified, while in runtime protection, vulnerabilities of the function or operation level will be exploited. Alfonso Rodríguez and others proposed a model-driven conversion framework [16]. Among them, by paying special attention to security requirements, information related to secure business processes is retrieved from the old information system. These security requirements are related to access control, auditing, integrity, and privacy. Security aspects are also crucial in a cloud environment. When migrating the legacy system to the cloud environment, the author has put forward a detailed investigation on security [17]. The author believes that when migrating old systems to the cloud, security or privacy aspects are usually ignored. With the help of four research questions, the author analyzed research publications related to safety. Security is also very important in real-time systems. The author

developed different techniques and a design-time evaluation framework for integrating security in real-time systems in [18]. The author further studied the measurement aspects and defined various measures to measure the effectiveness of the integration. Alain Bensoussan et al. proposed a model related to an intrusion detection system that can distinguish between legitimate and malicious traffic entering the company [19]. The author believes that when shocking attacks increase over time, the system's discriminative ability will decrease. Therefore, the proposed model proves the level of discrimination that companies should strive to achieve.

3 Security Modernization Framework

The Security Modernization Framework (SMF) uses two types of protection layers (i.e., database layer and application layer) to ensure that only legitimate access to old data and applications is made.

In Fig. 1, the two layers of security control are modeled. In the first layer (database layer), the proposed security methods are mainly related to data auditing. Database roles are related to user authorization and control. Database triggers audit the changes that users make to data in a specific transaction type. The DB log is a DBMS (i.e., Oracle) function used to record required information during database transactions. In the second layer (application layer), the DML recorder is used for data auditing in DML transactions, and the gateway controller manages vulnerabilities related to unauthorized access.

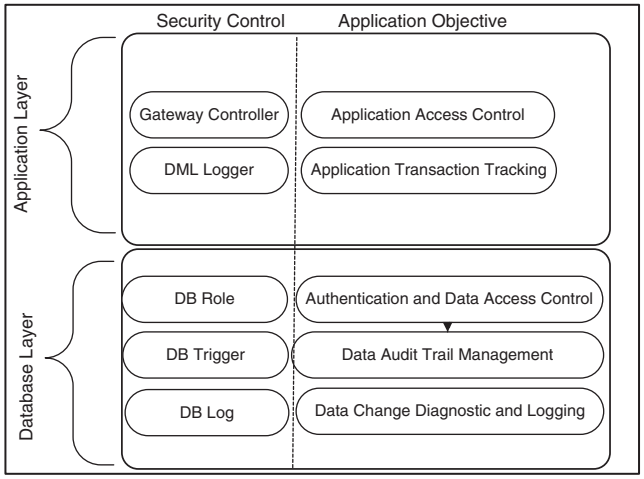

Figure 1: LIS's security modernization framework

3.1 Database Layer

As mentioned earlier, two layers of security control are shown in Fig. 1. In the database layer, security controls and methods are applied at the back-end or database level, so in this section, we discuss security methods and technologies implemented on the database level.

3.1.1 DB Role

Generally, the access control security mechanism of LIS is inappropriate. The permissions on different database objects are maintained chaotically. Usually, the direct access method is adopted; in which authorized users can directly access database objects. For example, Fig. 2 depicts a grid of unmanageable permissions between users U1, U2, and U3 on Tabs. T1 and T2. The simple scenario shown in Fig. 2 is difficult to manage and control. When tens or hundreds of database objects and users are involved, this method becomes more complicated and difficult to manage.

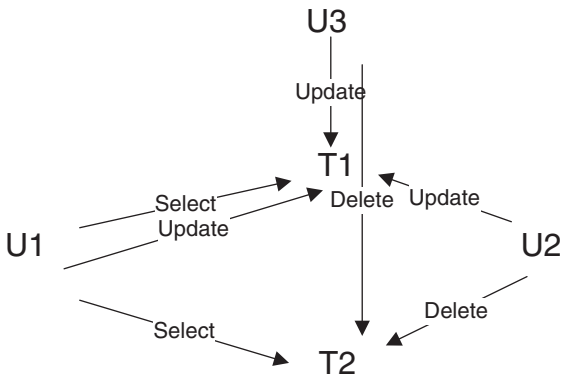

Figure 2: A mesh of unmanageable user rights

For DBAs, the complex user authorization method shown in Fig. 3 is even difficult, because managing and monitoring this type of grid is not an easy task. Therefore, using database roles for access control is a more appropriate, safe, and useful method. A role in a database is a set of permissions used to restrict user access to database objects (such as tables, views, procedures, functions, or packages, etc.). The DB role is a less complicated and easy way to group related permissions and privileges for better data and application access management. The following pseudocode illustrates setting database roles for different situations to enhance user access control mechanisms.

Scenario#1 (View only)

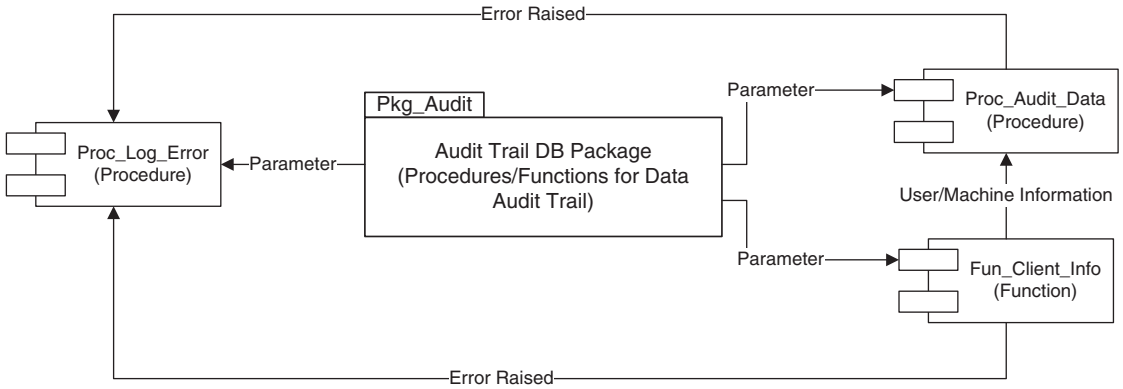

Figure 3: Outline of database package that maintains a data audit trail

a. Create role
b. Grant selection rights to the role on the required object(s)
c. Grant role to the relevant user(s)

Scenario#2 (View and Update)

a. Create role
b. Grant selection and update rights to the role of the required object(s).
c. Grant role to the relevant user(s)

Scenario#3 (View, Add/Remove, and Update)

a. Create role
b. Grant selection, update, and deletion rights to the role of the required object(s).
c. Grant role to the relevant user(s)

Under the above circumstances, access to the data in the existing LIS is restricted in three different ways. There may be other similar types of scenes involving multiple objects and users. This simple role-based method is more structured, efficient, and can protect valuable old data from unauthorized access by using logical permission groups.

3.1.2 DB Trigger

LIS usually does not have sufficient methods to maintain records of data changes. Therefore, if an unpleasant event related to a data security breach or unauthorized access occurs, there should not be any mechanism to detect or identify clues to changes made to sensitive data. Database triggers (database triggers) are a convenient way to capture a complete data audit trail. The following scenarios show how to maintain audit trails and control user access.

DB Level (Login/Logout Triggers)

The login/logout trigger is very important to track the entry and exit of users in LIS. Whenever a user enters or exits the application, the login and logout triggers are triggered separately. Through these two types of triggers, you can easily capture "who", "when" and "where" information.

Login Scenario

We create the following table to store the information of login.

Tab. 1 contains the basic information of the logged-in user, but more information can be maintained as needed. Let us describe the purpose of each column in the table.

Table 1: Database table (e.g., LOG_AUD_TAB), used to maintain user login/logout details

Column	Data type
USER_ID	VARCHAR2 (50)
S_ID	NUMBER (10)
HOST_NAME	VARCHAR2 (50)
LOGIN_DATE	DATE
LOGIN_TIME	VARCHAR2 (15)
LOGOUT_DATE	DATE
LOGOUT_TIME	VARCHAR2 (15)
TIME_SPENT	VARCHAR2 (15)

USER_ID	*is the person who establishes the connection*
S_ID	*is the current session id of login user*
HOST_NAME	*is the host/machine or location from where the session is established*
LOGIN_DATE	*is the date of login activity*
LOGIN_TIME	*is the time of login activity*
LOGOUT_DATE	*is the date of logout after work completion*

LOGOUT_TIME *is the time of logout after work completion*

TIME_SPENT *is the total work duration*

Now let's create a database-level login trigger that captures the information when a login event occurs (pseudocode is provided below)

/********** Pseudocode for login scenario **********/

/***/

1. Create a trigger for an event just after login
2. Get and insert the following data into the table LOG_AUD_TAB
    a. Login user
    b. Session
    c. The host name
    d. The date and time of login
3. Make the changes permanent

/********** Example trigger (oracle) **********/

/***/

```
CREATE OR REPLACE TRIGGER
TRG_LOGIN_TRAIL
AFTER LOGON ON DATABASE
BEGIN
    INSERT INTO LOG_AUD_TAB VALUES
    (USER,
    SYS_CONTEXT (‘USERENV’,'SESSIONID’),
    SYS_CONTEXT (‘USERENV’,'HOST’),
    SYSDATE,
    TO_CHAR (SYSDATE, ‘HH24: MI: SS’));
END;
```

Logout Scenario

Now we create a trigger for logout to capture the rest of the information:

/********** Pseudocode for logout scenario **********/

/***/

1. Create a Trigger for an event just before logout
2. Get and update the following data into the table LOG_AUD_TAB for the relevant user session
    a. The date and time of Logout
    b. Total duration of being logged in
3. Make the changes permanent

```
/******************** Example trigger (oracle) **********************/
/*********************************************************************/
        CREATE OR REPLACE TRIGGER
        TRG_LOGOUT_TRAIL
        BEFORE LOGOUT ON DATABASE
        BEGIN
        UPDATE
        LOG_AUD_TAB
        SET
          LOGOUT_DAY = SYSDATE,
          LOGOUT_TIME = TO_CHAR (SYSDATE, 'HH24: MI: SS'),
            TIME_SPENT= ROUND ((LOGOUT_DAY - LOGON_DAY)*1440)
        WHERE
          SYS_CONTEXT ('USERENV','SESSIONID') = SESSION_ID;
        END;
```

Table Level (Before/After Insert, Update, Delete Trigger)

Using table-level triggers to track data changes is another method that can be easily implemented in LIS to enhance the security of the system. Table-level triggers can maintain various types of operations performed by users during a transaction.

The following pseudocode maintains the audit trail of data change operations.

```
/**** Pseudocode to maintain data audit trail (back-end technique) ******/
/*********************************************************************/
```

1. Create a database package Pkg_Audit
2. Create procedure ***Proc_Audit_Trail*** for maintaining a data audit trail of who, when, what and how information using *old* and *new* data values.
3. Define a Function ***Fun_Client_Info*** with required arguments to capture and provide client details that are user, machine and source of transaction
4. Define a procedure ***Proc_Log_Error*** to record errors raised during the audit trail operation

3.1.3 DB Log

Most DBMSs usually provide a database log (DB Log) to maintain a detailed audit trail of all DML operations. This option is available in many DBMSs, and a database administrator (DBA) can use this option to maintain an audit trail. DB Log contains information related to a client computer, user, SQL statement, date, time, and operation type. However, one must use the DB Log utility with caution, because enabling this feature may adversely affect database performance.

3.2 Application Layer

After discussing the back-end security methods, we now discuss the front-end methods and techniques that can further enhance the security structure of LIS.

3.2.1 DML Logger

In order to enrich the data audit trail, it is necessary to apply few security controls at the application level. The DML operation performed by the user will be provided to the database trigger, which in turn inserts detailed information into the data audit trail table. The DB trigger discussed earlier has a limitation, that is, it is difficult to obtain the name of the relevant Application Server, especially in a three-tier environment. This means that not all types of information can be captured on the backend, so a small number of security methods need to be incorporated at the interface/application level so that a complete audit trail can be maintained. The following pseudocode helps to achieve the goal.

/********* Pseudocode to maintain data audit trail (front-end technique) ********/

/***/

a. Add the following six columns to the database table for storing the audit data:

   Entry_By Varchar2 (50),

   Entry_Date Date; – *The data type of the field should be set as DateTime at form-level*

   Updated_By Varchar2 (50),

   Updated_Date Date; – *The data type of the field should be set as DateTime at form-level*

   Client_Machine Varchar2 (100),

   Client_OS_User Varchar2 (100)

b. Using *Webutil* library, get the client machine name and operating system user in *global* variables (e.g., *G_Machine and G_OS*) via ***When_New_Form_Instance*** trigger of the relevant form in LIS.

c. Add code to ***Pre-Insert*** trigger of the relevant form so that:

   Entry_By captures the logged in data entry user;

   Entry_Date logs date and time of the insertion activity;

   Client_Machine takes the client computer/machine name stored in *G_Machine*;

   Client_OS_User takes the operating system user name stored in *G_OS;*

d. Add code to ***Pre-Update*** trigger of the same form, so that:

   Updated_By logs the user updating the data;

   Updated_Date logs date and time of update;

   Client_Machine takes the client computer/machine name stored in *G_Machine*;

   Client_OS_User takes the operating system user name stored in *G_OS;*

e. Add code to ***Pre-Delete*** trigger of the same form, so that:

   –Capture the old values before deletion

   Updated_By logs the user deleting the data;

   Updated_Date logs the date and time of data deletion;

   Client_Machine takes the client computer/machine name stored in *G_Machine*;

   Client_OS_User takes the operating system user name stored in *G_OS;*

3.2.2 Gateway Controller

The gateway controller is used to restrict unauthorized access and record illegal access attempts. This method uses two types of techniques.

Rule-Based Technique

In this technique, access rules are configured to associate authorized client computers with related application resources. Without defined rules, no user/client computer can access the application. In this technique, a separate table is maintained to store all associations between client computers and authorized application resources. Tab. 2 below shows the layout of the data store, which maintains the association rules between application resources and related client computers.

Table 2: Association rules for legitimate application access

Client_machine	OS_USER_ACCOUNT	App_resource
Client-computer1	Manager_01	HR system (HR-121)
Client-computer2	Advisor_04	Financial system (FS-233)
Client-computer1	Manager_03	Financial system (FS-234)
Client-computer1	Rep_02	Sales management (SM-400)

In Tab. 2, the OS_User_Account column contains the operating system user account that is authorized to use the client computer mentioned in the Client_Machine column, which can access the application resources given in the App_Resource column. For example, Manager_03 is a legitimate user account held by the financial manager. If the financial manager uses Client-Computer2 to access the financial system, it will be impossible, because only Client-Computer1 has the association rules of the defined account to access the relevant application resources.

Single Entry Point

A single point of entry is essential to a secure system. In this method, only one legal way to enter the system is defined and all other ways of entry are restricted.

As shown in Fig. 4, only one legal access point can be used to enter the LIS. This entry point is only available for users with authorized login credentials. Even if you call the corresponding URL directly, you cannot access LIS. Remote access to software resources from any process or function is not allowed. Similarly, software resources cannot be called by external systems.

The pseudocode given below is used to implement the *single entry point* method to control LIS access.

/****** Pseudocode for safe calling of software resources **********/

/***/

a. Store an encrypted key value in a table.

b. Define a global variable e.g., *G_Golden_Key* at the start of main MDI (Multiple Document Interface)

c. Decrypt the key value stored in the table and save it in *G_Golden_Key* at the start of MDI interface

d. Each time an SDI (Single Document Interface) is invoked, check for *G_Golden_Key=Key value stored in the table*.

e. If the result at step ‘d’ is TRUE, the user is allowed to access the SDI otherwise the access is denied

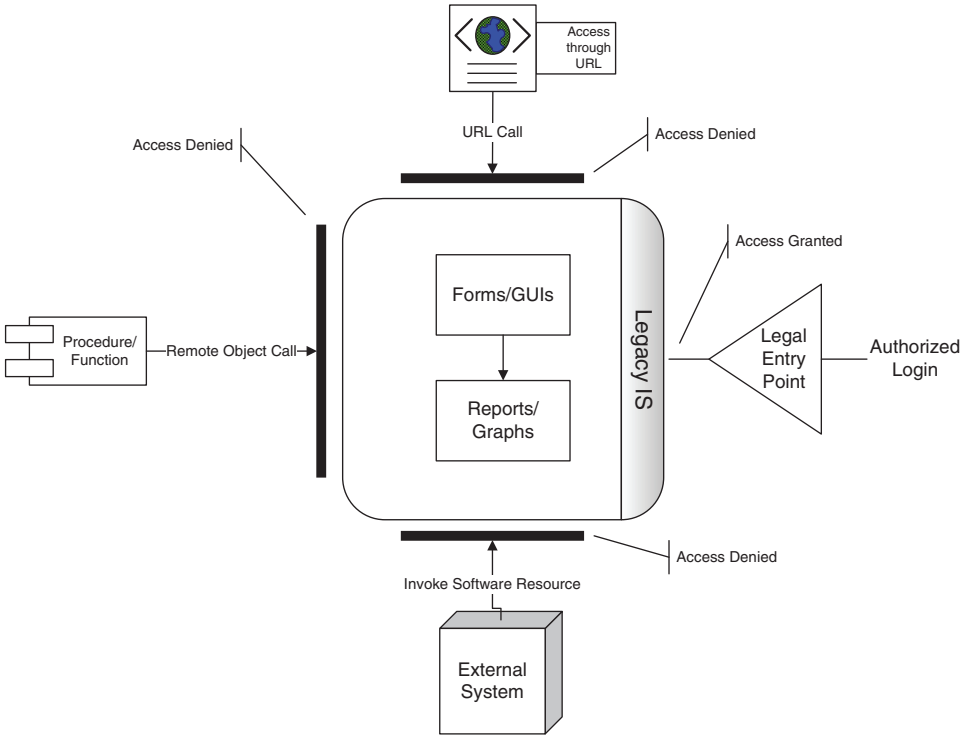

Figure 4: Single point of access for secure LIS

4 Implementation and Testing of SMF

The results were verified by implementing and testing all security controls using relevant security methods. To build trust, two types of testing methods were performed to verify the proposed framework.

4.1 Validation Questions

The security control and application methods were tested through related verification questions. As shown in Tab. 3, for each type of security control, verification questions are prepared in the form of test cases. These questions are selected from multiple functional areas.

After rigorous testing and verification of the results, the following output is given:

Table 3: Validation of SMF's controls

Security control	Testing method	Validation question	Results	Result status
DB trigger	Database transaction	Who did what and when?	Manager_01 updated the address of customer-03 on 08-11-2020 at 09:30:22	Qualified
DB log	Database transaction	Who issued Which SQL from client Client-Computer2 at 10:00 am on date 09-03-2020?	Advisor_04 issued “Select cutomer_name From customer where customer_no=‘cusomter-02’;”	Qualified

(Continued)

Table 3 (continued).

Security control	Testing method	Validation question	Results	Result status
DB role	Authorization control	Can Rep_02 of Sales Management see the product-wise revenues?	Rep_02 is unauthorized to access the product-wise revenues record	Passed
DML logger	DML operations	Which user deleted the salary account head using which interface?	Manager_03 deleted the salary account head using interface "FS-234"	Qualified
Gateway controller	Direct access	Can anyone directly access the interface "SM-400" without logging in to the application?	Cannot access the "SM-400" via URL	Passed

As shown in Tab. 3, all the descriptive questions have been successfully addressed, and the actual results are in line with the expected results. Those tests based on issues related to authorization checks were considered positive and were considered passed. In the end, we concluded that these simple security controls are very useful for enhancing the security and protection paradigm of LIS data and application in SMEs.

4.2 Vulnerability Testing

Vulnerability testing has been conducted thoroughly to ensure that SMF produces credible and accurate results. In this regard, two types of vulnerabilities are applied in the human resources, financial management, and administration domains, namely privilege vulnerabilities and transaction vulnerabilities. The results are given in Tab. 4.

Table 4: Test results to check transaction and privilege vulnerabilities

LIS	Security control	Vulnerability testing	
		Privilege	Transaction
Human resource	DB trigger	✗	✓
	DB log	✗	✓
	DB role	✓	✗
	DML logger	✗	✓
	Gateway controller	✓	✗
Financial management	DB trigger	✓	✓
	DB log	✗	✓
	DB role	✓	✓
	DML logger	✗	✓
	Gateway controller	✓	✓

(Continued)

Table 4 (continued).

LIS	Security control	Vulnerability testing	
		Privilege	Transaction
Administration	DB trigger	✗	✓
	DB log	✗	✗
	DB role	✓	✗
	DML logger	✗	✓
	Gateway controller	✓	✗

After applying relevant security controls in the corresponding LIS, the results of the vulnerability test are displayed in Tab. 4. The results plotted in the table show that all the security controls of SMF produced excellent results. The symbol "✓" indicates that the corresponding security control has been applied and the related vulnerabilities have been successfully deleted, while "✗" indicates that the corresponding security control has not been applied. The drawn results show that all security control measures have the ability to eliminate related vulnerabilities. The LIS of these three functional domains was tested under relevant security control and produced remarkable results.

5 Conclusion

Security is an important attribute of quality information systems. Unauthorized or malicious attacks to access valuable data and information can cause considerable losses to enterprises. Unfortunately, organizations often lack this vital quality attribute in their LIS. In order to protect their legacy data and applications, large enterprises usually choose a modern and complete LIS infrastructure. However, for small and medium enterprises (SMEs), the modernization of the entire legacy system is not an easy task, because it requires a lot of cost, a long time, and a lot of work. Therefore, instead of transforming a complete LIS into a new modern system, we propose a Security Modernization Framework (SMF), which includes a set of simple security controls that can enhance the security paradigm of LIS. SMF has two layers of security control, the first layer is the database layer, and the second layer is the application layer. The results show that with minimal effort and time, the security of LIS has been significantly improved, and existing vulnerabilities have been greatly reduced. By improving the security functions of existing information assets with minimal effort and the lowest cost, SMEs can benefit from SMF. As the example shows, SMF is implemented in the context of oracle technology; however, the pseudocode also illustrates methods that are easy to refer to in other technologies.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Bisbal, D. Lawless, B. Wu and J. Grimson, "Legacy information system migration: A brief review of problems, solutions and research issues," *IEEE Software*, vol. 16, no. 6, pp. 103–111, 1999.

[2] S. M. Hussain, S. N. Bhatti and M. F. U. Rasool, "Legacy system and ways of its evolution," in *Proc. ComTech*, Rawalpindi, Pakistan, pp. 56–59, 2017.

[3] R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen and J. Hage, "How do professionals perceive legacy systems and software modernization?," in *Proc. ICSE*, Hyderabad, India, no. 1, pp. 36–47, 2014.

[4] B. Y. Alkazemi, M. K. Nour and A. Q. Meelud, "Towards a framework to assess legacy systems," in *Proc. IEEE SMC*, Manchester, UK, pp. 924–928, 2013.

[5] B. Potter and G. McGraw, "Software security testing," *EEE Security & Privacy*, vol. 2, no. 5, pp. 81–85, 2004.

[6] K. A. Barton, G. Tejay, M. Lane and S. Terrell, "Information system security commitment: A study of external influences on senior management," *Computers & Security*, vol. 59, pp. 9–25, 2016.

[7] A. Ključnikov, L. Mura and D. Sklenár, "Information security management in SMEs: Factors of success," *Entrepreneurship and Sustainability Issues*, vol. 6, no. 4, pp. 2081–2094, 2019.

[8] M. Hallová, P. Polakovič, E. Šilerová and I. Slováková, "Data Protection and Security in SMEs under Enterprise Infrastructure," *AGRIS on-line Papers in Economics and Informatics*, vol. 11, no. 1, pp. 27–33, 2019.

[9] R. Khadka, A. Saeidi, S. Jansen, J. Hage and G. P. Haas, "Migrating a large scale legacy application to SOA: Challenges and lessons learned," in *Proc. WCRE*, Koblenz, Germany, pp. 425–432, 2013.

[10] X. Li and Y. Xue, "A survey on server-side approaches to securing web applications," *ACM Computing Surveys*, vol. 46, no. 4,. pp. 1–42, 2014.

[11] F. Fui, H. Nah, S. Delgado and F. Fui-Hoon Nah, "Critical success factors for enterprise resource planning implementation and upgrade," *Journal of Computer Information Systems*, vol. 46, no. 5, pp. 99–113, 2006.

[12] B. Paradauskas and A. Laurikaitis, "Business knowledge extraction from legacy information systems," *Information Technology & Control*, vol. 35, no. 3, pp. 214–221, 2006.

[13] R. Sahandi, A. Alkhalil and J. Opara-Martins, "IFIP AICT 380-SMEs' perception of cloud computing: Potential and security," in *Proc. WCVE*, Springer, Berlin, Heidelberg, vol. 380, pp. 186–195, 2012.

[14] R. Al-Ali, P. Hnetynka, J. Havlik, V. Krivka, R. Heinrich *et al.*, "Dynamic security rules for legacy systems," in *Proc. ACM ECSA*, Paris, France, vol. 2, pp. 277–284, 2019.

[15] A. Messaoud Benantar, "Method and system for public-keybased secure authentication to distributed legacy applications," United States patent application US 09/821, 079, 2002.

[16] A. Rodríguez and C. Angélica, "Towards framework definition to obtain secure business process from legacy information systems," in *Proc. MESDQS*, Hong Kong, China, pp. 17–24, 2009.

[17] L. M. Alcañiz, D. G. Rosado, D. Mellado and E. Fernández-Medina, "Security in legacy systems migration to the cloud: A systematic mapping study," in *Proc. WOSIS-ICEIS*, Lisbon, Portugal, pp. 26–37, 2014.

[18] M. Hasan, "Integrating security into legacy real-time cyber-physical systems," Ph.D. dissertation, Graduate College of the University of Illinois at Urbana-Champaign, 2020.

[19] A. Bensoussan, V. Mookerjee and W. T. Yue, "Managing information system security under continuous and abrupt deterioration," *Production and Operations Management*, vol. 29, no. 8, pp. 1894–1917, 2020.