Security Analysis and Implementation of Web-based Telemedicine Services with a Four-tier Architecture

Amiya K. Maji, Arpita Mukhoty, Arun K. Majumdar, Jayanta Mukhopadhyay, Shamik Sural, Soubhik Paul, Bandana Majumdar
Indian Institute of Technology
Kharagpur, India
{amiya, amukhoty, akmj, jay, shamik, spaul, m_bandana}@cse.iitkgp.ernet.in

Abstract—**Security of Telemedicine applications is not often given adequate importance by the developers and healthcare administrators primarily to reduce cost. Though some security safeguards are employed by these applications to comply with existing medical data security and privacy regulations, these are not adequate in today's context. Moreover, in a web-based application environment not only the data but also the application itself is vulnerable to attackers. Keeping these concerns in mind, we present the design of a web-based, four-tier Telemedicine System named iMedik which is accessible over desktops as well as handheld devices. We have illustrated how the proposed system differs from existing three-tier web applications. The compliance status of the application with HIPAA Security Guidelines has also been noted. The security measures described in our approach look into the four-tier architecture from an attacker's viewpoint and present a simple road map for developing secure e-health application with anywhere, anytime availability.**

Keywords—multi-tier; telemedicine; vulnerability analysis; e-health; web based

I. Introduction

Internet, due to its anywhere anytime availability, has influenced the trends in application development to a great extent over the last decade. Organizations are putting more emphasis on developing web-based systems because of the wider availability and lesser cost of these applications. Research in the field of e-Healthcare applications has been no exception. Support from various government and private institutions has led to the genesis of a plethora of web-based medical software during a short span of time. A literature survey on the current trends in e-Healthcare research shows several projects dedicated to the development of web-based applications to support enhanced patient care. While several researchers have presented Internet based Electronic Medical Record (EMR) systems [1, 2], development of web-based applications for emergency care and building collaborative care environments for improved healthcare services [3–5] have also been active areas of study. Researches in the direction of healthcare delivery at home [6, 7] and online monitoring of patients' vital statistics [8] have gained momentum in recent times. Surprisingly, security considerations in these applications are very often found inadequate for the risks prevalent in the Internet environment. A great majority of the existing web-based e-Healthcare softwares incorporate transmission security as the sole means of protecting medical data. Some of the approaches also use digital certificates in conjunction with SSL for user authentication. Very few of the articles discuss about other aspects of security like access control, auditing and session management [3, 6, 9]. Further-more, none of these works have been found to contain discussions on *application security*.

In this paper we present the architecture of a web-based four-tier telemedicine system named *iMedik* which has been developed with a major emphasis on security. Our work extends the existing three-tier application architecture [6, 7, 9] to incorporate an additional layer of security. In this architecture we are able to protect not only the medical information but also the application components from hackers. We have illustrated how the proposed four-tier architecture imparts security, flexibility and robustness into the application. We also present an analysis of the security of the proposed system in the context of some common web-application vulnerabilities. Emphasis on application security has been given due to the recent rise in hacking incidents at the web application level. Thus, our approach looks into the four-tier architecture from an attacker's viewpoint and presents a simple road map for developing secure e-health application.

The rest of the paper is organized as follows. Section II gives us an overview of the architecture of proposed system followed by a description of the security module and its analysis in Section III. Implementation details of the system are highlighted in Section IV. In the succeeding section, a comparison of the proposed architecture with the three-tier architecture has been presented. Finally Section VI concludes the paper by highlighting ongoing works in this direction.

II. System Architecture

Developed with the requirements of security, flexibility, robustness and availability, iMedik enhances existing three-tier web application development architecture [10, 11] by introduction of an additional layer of security. During our development phase it was observed that the generic three-tier architecture consisting of Database Tier, Application Server and Client Tier [Refer Fig. 1] is an ideal development model where the different tiers reside within the perimeter firewall of an organization. In the Internet environment, the Client is typically replaced by a Web Browser and the Application Server is placed behind a firewall with the introduction of a

This project is supported by grant No. 1 (23)/2006-ME&TMD of the Ministry of Communications and Information Technology, Govt. of India.

public HTTP Server [12, 13]. However, in both these configurations, some portion of the application logic resides on the public server outside the firewall and is vulnerable to hacking attempts. The HTTP Server, which resides in the Demilitarized Zone (DMZ), consists of presentation codes written as scripts (asp/jsp/php/perl etc.). If this layer is compromised, the hackers can view the script codes and learn about sensitive implementation details. Moreover, in such situations, the security system at HTTP Server may be bypassed to access arbitrary components at the Application Server.

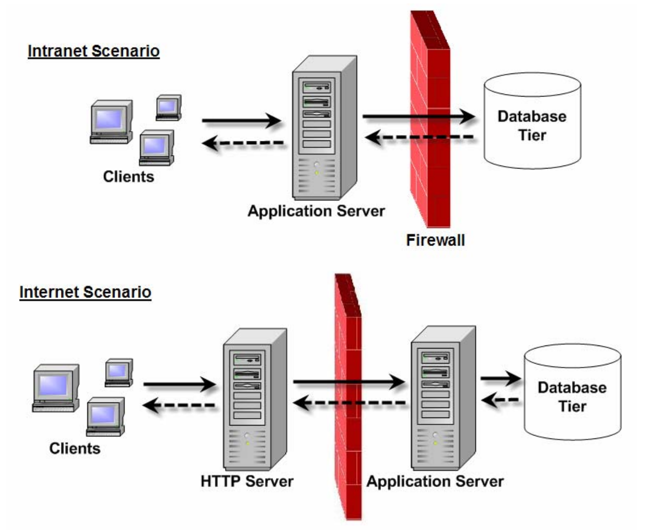


Figure 1. Typical web-application deployment scenario

A. The Four-tier Architecture:

To overcome the shortcomings of the three-tier architecture, we have customized the generic architecture to a four-tier model composed of Database Layer, Business Logic Layer, Presentation Layer and Web Proxy Layer [Refer Fig. 2]. As the client in our case is a standard web browser, it is not considered as a component layer of the iMedik system. The ***Database Layer*** is the lowest layer of the application. Since it is the most sensitive segment of the application, database is always protected by a firewall. Connection to database can only be established by Business Logic Components. No other machine or application is allowed to communicate with the database directly. The second layer in our proposed architecture is the ***Business Logic Layer*** which contains most of the application logic. This layer is essentially the core of the application. It performs all the database operations and carries out computations on the fetched data. The layer above the Business Logic Layer is termed as the ***Presentation Layer***. It intercepts all requests and responses to and from the Business Logic Layer. Depending on the user request, this layer performs filtering and formatting operations on the input and output respectively. All these layers reside behind a ***Firewall*** and hence can be termed as *Private Layers* or *Internal Layers*. The firewall protects the Internal Layers from hacker attacks.

The fourth and outermost layer of the application which distinguishes our approach from generic three-tier applications is the **Web Proxy Layer**. This layer acts as the point of entry to the proposed system. The Web Proxy resides in the Demilitarized Zone (DMZ) outside the firewall and intercepts all the http requests from users and forwards them to the Internal Layers. The Web Proxy Layer also provides a single point for validating user requests. It plays important role in maintaining user sessions. Moreover, the presence of Web Proxy allows us to hide all the application components behind the firewall.

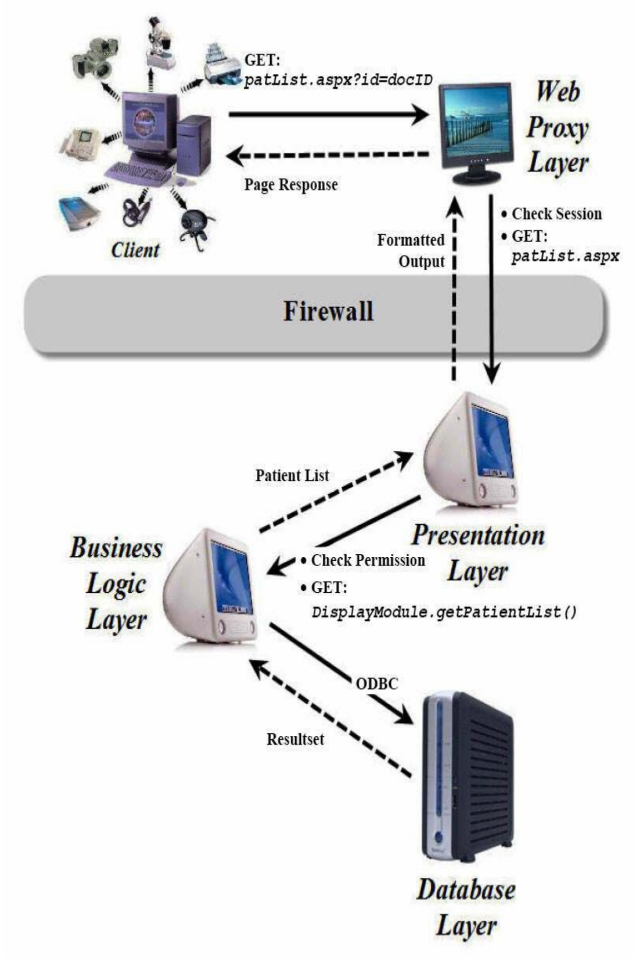


Figure 2. Layout of the proposed system.

We now illustrate the functioning of the application by analyzing the processing sequence of a sample request [Refer Fig. 2]. Assume that a Doctor D_1 wants to view the list of patients under him. So he sends a request for the page "*patList.aspx*" to the system. The request is first intercepted by the Web Proxy which verifies whether D_1 is already authenticated to the system. If so, the request is passed to the Presentation Layer. The Presentation Layer now validates the user inputs (form data, cookies, querystring etc.) and then calls the appropriate Business Logic Component, say *DisplayModule.getPatientList()*, for retrieving the patient list from the database with *doctorID* as an argument. The Business Logic Layer initially checks the access permission and then returns the result of the database query to the Presentation

Layer. The result is now formatted as html output and returned to the Web Proxy by the Presentation Layer. Finally the Web Proxy forwards the response to D_1. It is to be mentioned that all communications between the Client and the Web Proxy and between the Web Proxy and the Presentation Layer are protected by SSL.

B. Advantages:

The four-tier architecture discussed above presented the developers with a significant set of advantages. These are –

1) Enhanced Security: It increases the overall security of the system by distributing the application components over physically disjoint machines. Moreover, it incorporates component security in the system. Details on other aspects of security will be presented in the next section.

2) Cleaner Segregation of Application Logic and Presentation Code: This architecture segregates the Presentation Logic from Business Logic, thus increasing the manageability of the application.

3) Flexibility: The four-tier architecture also increases the flexibility of the system. Modifications in the Presentation Layer and the Web Proxy Layer is independent of the Business Logic Layer. We can have multiple presentation code communicating with the same Business Logic Layer thereby allowing seamless integration of various user access modalities. Such integration is a significant step towards the development of a pervasive healthcare system.

4) Scalability: It increases the scalability of the overall system. If all the layers are placed on a single computer, the application will be able to support lesser number of concurrent users than it can in the four-tier configuration.

C. Wireless Access:

Since handheld devices (PDA and Cellular Phones) have very small screen resolution we have developed a special module in iMedik for access over these devices. The *Wireless Rendering Module* is a part of the Presentation Layer and it uses the same Business Logic Components as the Desktop Presentation Module. The wireless module is a direct example of the flexibility of the proposed system and an initial step for supporting pervasive healthcare services. Fig. 3 displays the architecture of iMedik in conjunction with the Wireless Rendering Module. The equivalence sign in the picture depicts that the presentation modules for desktop and wireless access may be hosted on the same server computer.

III. Security System Design and Analysis

To protect an application effectively it is necessary for the designers and security administrators to think like a hacker. Otherwise, an application may be easily compromised due to flaws in the application code. Hence we present the description of the security module of our proposed system in this section by considering common vulnerabilities and then discussing about the countermeasures incorporated into the system to address the same. In a web-based application such as ours, security is essentially dependent on three components namely Host Security, Network Security and Application Security. We shall mostly focus our attention to the *Application Security* aspects in this paper.

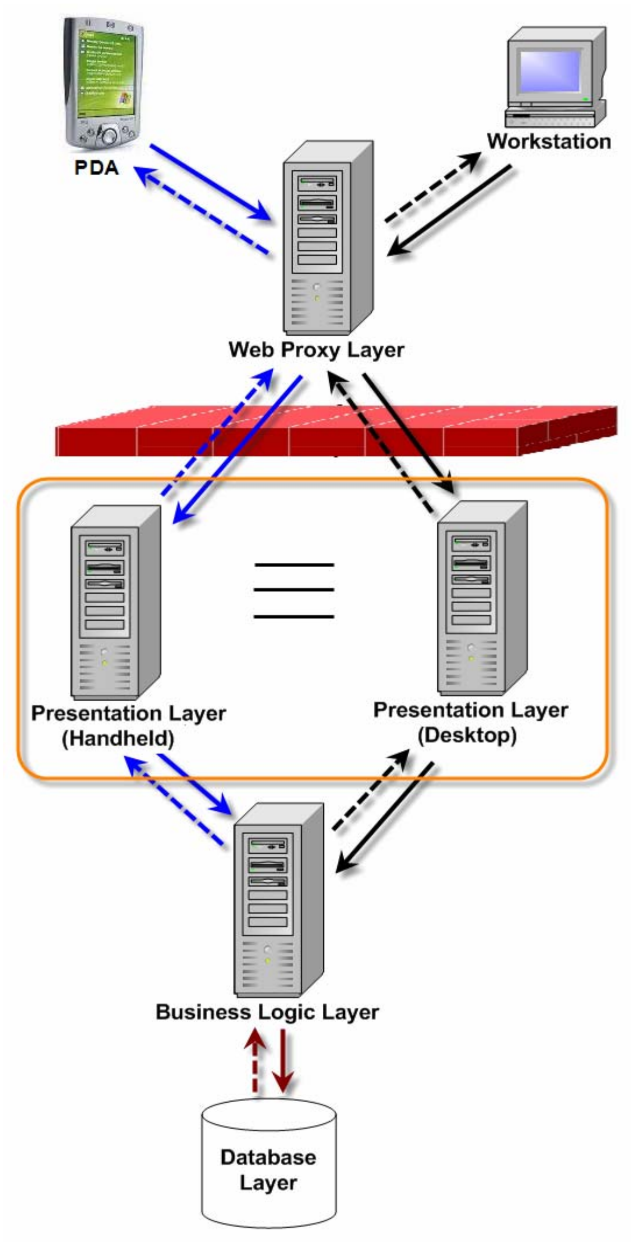


Figure 3. Access over wireless devices

A. Protection Against Web Application Vulnerabilities

With the advent of new web development technologies, application level vulnerabilities are gradually increasing in number. Presence of multiple platforms and browsers allows these vulnerabilities to be exploited by an array of attack vectors. Protection against all known web application vulnerabilities is thus an arduous job. During the development of iMedik, we considered a limited set of common web application vulnerabilities for designing the security module. The vulnerability list was prepared by consulting the OWASP Top Ten [14] list, Microsoft STRIDE [15] model and some other articles [16–19]. Table I lists the set of vulnerabilities that have been addressed in the proposed system.

TABLE I. COMMON WEB APPLICATION VULNERABILITIES

Sl No.	Vulnerability Name
V01	Cross-Site Scripting (XSS)
V02	Injection Flaws
V03	Authentication and Session Management
V04	Insecure Communication
V05	Insecure Direct Object Reference
V06	Information Leakage
V07	Insecure Cryptographic Storage
V08	Failure to Restrict URL Access
V09	Error Handling
V10	Auditing and Logging
V11	Buffer Overflow
V12	Denial of Service (DOS)
V13	Broken Caching and Reuse
V14	Logical Attacks

Among the listed vulnerabilities, V01, V02 and V11 normally arise due to improper user inputs and faulty input validation routines. On the other hand, V03, V05, V08 and V10 are results of poor designing. Improper configuration of the web application leads to V06, V07 and V09. Vulnerabilities V12–V14 can be mitigated by careful design and implementation, whereas, V04 requires the developer to use some existing network security protocols like SSL/TLS. Detailed description about these vulnerabilities can be found at [14–16, 20].

1) Security Functionality at Various Layers: In this section we present the security measures incorporated in each of the layers to handle the vulnerabilities mentioned above. It is followed by a discussion on the reasons for segregating the security routines into four layers. We also present a brief example illustrating how the mentioned security routines are enforced when a particular webpage is requested.

a) Web Proxy Layer:

- It plays important role in *managing and verifying user sessions*. When a user requests for a particular webpage, this layers checks if the session specified by the session cookie is valid. A request is passed to the Internal Layers only if the session is valid. Apart from preventing cookie replay attack, it also saves the resources of Internal Layers by blocking invalid user requests.
- The system manages two different session objects for a user. The Web Proxy *maps the local session to the internal session* (i.e. session at Presentation Layer). In this scheme the reference of the internal session is never revealed to the user.
- Limits simultaneous user connections to the server thereby *reducing the risk of DOS attacks*.
- Restricts user access to known file extensions and a specific set of resource locations. Thus, Web Proxy Layer *prevents execution of arbitrary files* and blocks path traversal attacks.
- The presence of this layer allows the developer to mask the names of the internal objects (i.e. internal urls). It incorporates *indirection for various resources* by default.
- Depending on the type of content being requested by the user (static/dynamic or sensitive/non-sensitive) this layer can mark the response page as cacheable or not. It provides *protection against broken caching and reuse vulnerability*.
- When a user uploads a file to the system (e.g. patient image), this layer checks the uploaded file for malicious contents. It thus *protects the Internal Layers from being infected*.

b) Presentation Layer:

- The most important security functionality of the Presentation Layer is *input validation*. This layer checks user inputs for data type, maximum and minimum size, allowable character set and regular expressions. In case an input violates these positive specifications, it is rejected.
- This layer also *contains filters for XSS and SQL Injection*. All user input is verified on the basis of known XSS and SQL Injection signatures. Computation on the user inputs is done only after filtering.
- Presentation Layer plays important role in *performing URL access control* in the system. It maintains a *role-based access control (RBAC)* list specifying user permissions and allows or denies requests to different resources on the server.
- It *maintains the user's session information* when a user is logged in. This session information is internal to the server and it's not sent to the client. The session is expired after a specified period of inactivity or log out by the user, whichever is earlier.
- *Hidden form values and querystring values are encrypted* by the Presentation Layer using a session key. The encryption key is generated by the .Net framework and saved in machine key store during session initiation. It is deleted when the session expires. This scheme protects the application from parameter tampering attacks to a large degree.
- Before sending the output to the user, this layer *performs output encoding* of the untrusted data

elements. Thus the system is protected against XSS attacks.

- The Presentation Layer also limits resources used by different users, thereby *mitigating risks of DOS attacks*.
- Moreover, automated submissions of forms for important user activities like prescription writing, patient referral, and image upload etc. are blocked by *employing graphical confirmation schemes* [21].

c) Business Logic Layer:

- This layer *performs a component level access control* for user requests. When a particular webpage is requested, the Business Logic Layer checks which components are being used by the user. It then takes necessary access decisions depending on the policy.
- *Hashing of user passwords and encryption of sensitive data* in the database are other functionality of the Business Logic Layer. It also incorporates strong password enforcement routines into the system.

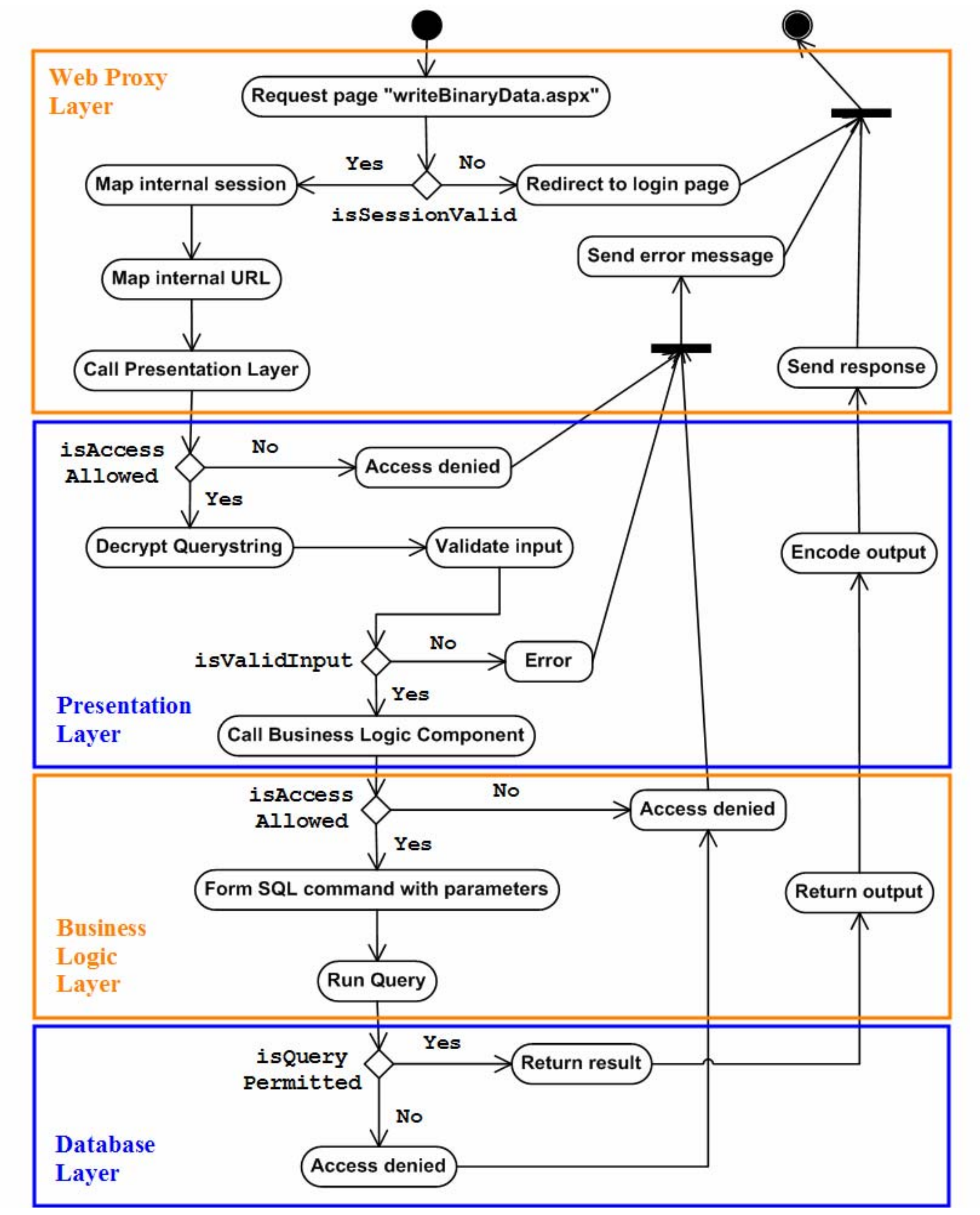


Figure 4. Security functionality at various layers

- While connecting to the database, this layer always *uses low-privileged database accounts* depending on the user's role. Moreover, the use of *parameterized SQL* eliminates risks of SQL Injection attacks.
- Blocks user accounts after a maximum number of failed login attempts. Thus it protects the system against *dictionary attack on user passwords*.
- Since this layer is responsible for encryption of sensitive data, it secures the encryption key by splitting the key and storing it in machine key stores. Thus the system *protects itself against insecure cryptographic storage vulnerability*.
- Furthermore, the Business Logic Layer is configured properly to restrict access to configuration files, policy files, log files, key stores etc.

d) Database Layer:

- The Database Layer enforces confidentiality and integrity of patient records by allowing *access to different database tables and views based on the requesting users' roles*.

e) Common Functionality: Apart from the functionality mentioned above, each of the first three layes also have some common security features which are very essential for overall security of the system.

- *Logging:* Each of the layers in our proposed system contains a logging module which helps the administrators to analyze unusual user behaviours.
- *Error Handling:* By default the layers are configured to send only custom error messages in case of exception. This prevents the attacker from getting sensitive information about the implementation details of the system. Moreover, the Web Proxy Layer always traps errors from the Internal Layers and sends "200 OK" message to the client, thereby masking error conditions.
- *Communication Security:* The communication between user client and the Web Proxy Layer is done using SSL to prevent network eavesdropping. Messaging between different layers in iMedik is also SSL protected.
- *Server Configuration:* Each of the layers of iMedik is configured to release minimal information about the server platform and block arbitrary path traversal.

Let us not illustrate how the security functionality are enforced at various layers when a webpage is requested [Refer Fig. 4]. Suppose a doctor wants to view a medical image of a particular patient. When he clicks on the image link a request is generated for the webpage

"*https://website/displayModule/writeBinaryData.aspx*".

This webpage takes five inputs: the patientId, image type, image serial no., the date on which the image was entered and the doctorId. Among these, the first four inputs are supplied as querystring parameters whereas the doctorId is retrieved from the current session object. The querystring for the Url request looks as "*id=VHOS1208070001&ty=BLD&sl=01&dt=12/8/2007*". However, the security module in the Presentation Layer encrypts the querystring to prevent it from being modified. Thus the encrypted string as displayed to the user is

"*https://website/displayModule/writeBinaryData.aspx?ShUyU6zqzx5yCLH%2bq...TpJ1mIN%2b4%3d*".

This request is first intercepted by the Web proxy Layer which verifies the user's session, maps the external session identifier with the internal session, converts the url mentioned above to the internal url and then sends the request to the Presentation Layer. The Presentation Layer first checks the user's access permission for this particular page. Then the querystring is decrypted with the session key. After verifying the input parameters, a request is made to the Businees Logic Components for retrieving the specified image from the database. The Business Logic Layer checks the permission of the user on the requested component and then issues a database query with the input values as SQL parameters. Finally the image is displayed only if the requesting doctor has viewing permission on the patient records. During processing each of the layers maintain log of the objects being accessed. Error situations, if generated, are also handled appropriately.

2) Advantages of the Four-tier Security Module: The security system of iMedik, which is distributed over the different layers, presents us with a rich set of functionality and greater advantages over typical three-tier systems. The four-tier security module presented in this paper works as a composition of the functionality embedded at various layers. Removing the security modules of any layer may render the application vulnerable to certain types of attacks. The strength of the four-tier security module derives from the following:

a) Redundancy: Due to the four-tier architecture some of the security modules could be replicated at various layers (e.g. access control at presentation and business logic layer, session management at web proxy and presentation layer etc.). As a result of this, if an attacker escapes security checks at a particular layer, the system can still block the request at deeper level.

b) Component Security: Introduction of the Web Proxy allows a developer to hide all the components of the application including the Presentation Logic and the Business Logic behind the firewall. The Web Proxy which is exposed in the DMZ contains only minor Validation Logic. *This ensures that even if the Web Proxy is compromised, the core of the application will still be unaffected.* Moreover the communication from the External Layer to the Internal Layers is authenticated by client certificate. Thus any malicious program running on Web Proxy must have access to a valid client certificate to establish connection with Presentation Layer.

c) Sandboxing: The Web Proxy Layer, which can intercept all requests (responses) coming to (from) the system, also acts as a virtual sandbox for the entire application. It can filter any of the requests or responses depending on the security requirements.

d) Ease of Implementation: The four-tier architecture also gives us a great degree of flexibility for implementing many of the security modules. E.g. we can implement error handling very easily in this system by trapping all exceptions at the Web Proxy Layer. This scheme hides error conditions from the user even if the Internal Layers do not have an error handling module in place.

e) Performance: Since the security functionality are distributed over four-layers the overhead at each of the servers is less. If the entire security code is placed on a single machine (say Business Logic Layer), the response time for that server may increase leading to a degradation of performance for the entire system. Moreover, correct functioning of that layer becomes crucial for security of the system.

3) Compliance with Security and Privcy Regulations: Various countries in the world have developed their own set of regulations for protection of individuals' privacy and their healthcare records. Among these the HIPAA regulations [22, 23] set by the US Department of Health and Human Services, the EU Directive on Personal Data Protection [24], European Standards and Guidance on Privacy and Confidentiality in Healthcare [25] and the Personal Information Protection and Electronic Documents Act [26] in Canada are notable. Due to the lack of security and privacy regulations for e-Healthcare data in India we decided to assess iMedik against HIPAA Security Standards. HIPAA was chosen as "a reference guideline" for medical data security because of the wide-scale availability of HIPAA related documents over the Internet. During the development of our application we followed some of these guidelines and implemented the 'Required' as well as the 'Addressable' safeguards. Our work mostly pertains to the "Technical Safeguards" section of the "*HIPAA Security Standards*" which specifies Access Control, Audit Control, Integrity, Person or Entity Authentication and Transmission Security as some of the requirements for achieving HIPAA compliance [23]. On the basis of the preceding analysis it can be clearly stated that our proposed system meets all these requirements. Moreover, to ensure the integrity of medical records stored in the database, users are not allowed to modify or delete any data without administrator's approval.

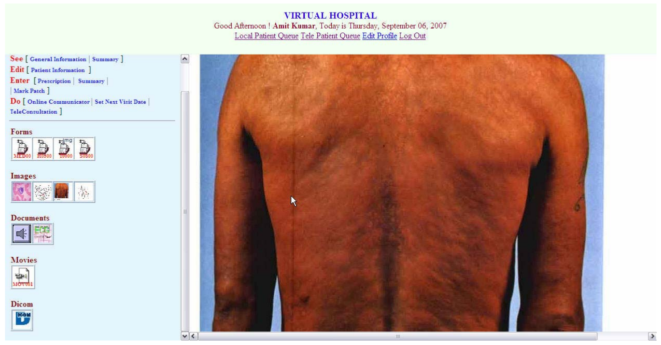
IV. IMPLEMENTATION

iMedik, our prototype four-tier telemedicine system, has been developed using Microsoft .Net framework. The medical database has been hosted in an MS SQL Server. It contains various clinical data in well structured format. All the other layers of the application are hosted in IIS Servers. These layers have been deployed on physically separate machines. The Web Proxy Layer consists of executables for accepting http requests whereas the Presentation Layer is composed of Asp, Html and Asp .Net pages. The Business Logic Layer has been developed in C# and it contains .Net Remoting objects. The user interface is completely browser based and hence the user need not install any separate software for using our application. Communication between the Client and the Web Proxy Layer is done using HTTP(S) messages. The Presentation Layer also processes HTTP(S) requests. The communication between the Presentation Layer and the Business Logic Layer is SOAP formatted. It is to be noted that SSL alone is inadequate to support non-repudiation of user activities in an application. Our current system incorporates non-repudiation by monitoring and analysis of log files. However, we are in the process of implementing existing XML-Security based non-repudiation schemes into our system.

Apart from the browser-based user interface and the security functionality, iMedik also has the following features:

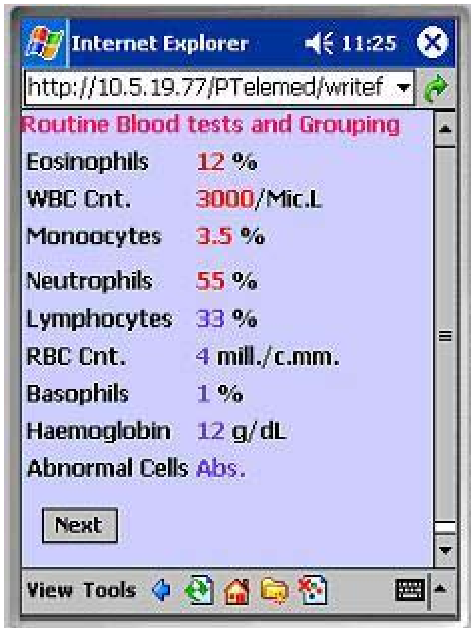
- It complies with the Electronic Patient Record (EPR) standards draft proposed by National Task Force for Telemedicine Standards, Ministry of Communication and Information Technology, Govt. of India. [27]
- Supports uploading and display of textual (txt, doc, pdf etc.), multimedia (image, audio, video, Dicom) as well as graphical data (ecg data).
- Incorporates functionality for visualization and annotation of human profiles and medical images to facilitate medical discussions.
- Supports creation of *electronic whiteboards* for online consultation among multiple doctors.
- The system includes utilities for Patient Information Backup and Restore.
- Supports generation and display of patient statistics as part of the administrative module.

During implementation of the system, the technical guidelines for secure web application development described in [17] were consulted. The article presented us with a brief overview of secure application development methodology and a checklist for verifying the same. At different phases during the implementation the system was tested for logical correctness as well as for certain types of vulnerabilities. The Nessus tool [28] was used extensively for scanning the servers for open ports and related vulnerabilities. Appropriate corrective measures were taken according to the scan results. We also used Paros Proxy [29] for testing the application for different types of Input Injection and Parameter Manipulation attacks.


VIRTUAL HOSPITAL
Good Afternoon ! **Amit Kumar**, Today is Thursday, September 06, 2007
Local Patient Queue Tele Patient Queue Edit Profile Log Out

See [General Information | Summary]
Edit [Patient Information]
Enter [Prescription | Summary | | Mark Patch]
Do [Online Communicator | Set Next Visit Date | TeleConsultation]

Figure 5. Dicom viewer utility in iMedik



http://10.5.19.77/PTelemed/writef

Routine Blood tests and Grouping

Eosinophils	12 %
WBC Cnt.	3000/Mic.L
Monoocytes	3.5 %
Neutrophils	55 %
Lymphocytes	33 %
RBC Cnt.	4 mill./c.mm.
Basophils	1 %
Haemoglobin	12 g/dL
Abnormal Cells	Abs.

Figure 6. Blood test report displayed on PDA

Fig. 5 illustrates the Dicom Viewer tool of iMedik whereas Fig. 6 shows a patient's blood test report as displayed on a PDA. Note that in Fig. 6 common words have been abbreviated to fit in the small screen size of PDA.

V. Comparison with Three-tier Systems

It is to be emphasized that the proposed four-tier architecture presents several advantages over existing three-teir applications. *Firstly*, let us consider the case of a typical three-tier web application where the application logic / presentation logic is hosted on a public computer [Refer Fig. 1]. This computer communicates with an Internal Database which is protected by a firewall. Since the web server is always hosted in the DMZ it is vulnerable to hacking attempts. If this server is compromised, the database is no longer hidden to the attacker. The attacker can easily modify the application components to perform arbitrary operations on the database. Moreover if the web application consists of codes written in the scripting languages the entire application logic is exposed to the hacker.

In comparison, the proposed four-tier system hosts the application components securely behind a firewall. The Web Proxy Layer consists of only executable codes. Hence the application logic is never visible to the attacker. Even if the Web Proxy Layer is compromised, the medical database is never directly accessible to the attacker. The database can only be accessed using the Business Logic Components. Thus the difficulty of hacking the medical database increases manifold.

Secondly, in a three-tier system, access control decisions are usually taken at the Application Server as this resides in the private network and is considered hack proof. However, in the proposed system we can have two-level access control mechanism (on physically separate machines) behind the firewall which is not possible in three-tier configurations. This two-level scheme ensures that a request need not travel as far as the Business Logic Layer before an access violation is triggered.

Thirdly, the four-tier architecture gives us cleaner segregation of presentation logic and business logic. Separate modalities for access over various devices can be seamlessly integrated with a single business logic server. The decision about which module at the Presentation Layer should be accessed can be taken intelligently by the Web Proxy Layer by monitoring the client resolution. In a three-tier system, this segregation is typically done by assigning separate domains (sub-domains) for different modalities.

Fourthly, the proposed architecture presents a custom-built, secure gateway for the entire application in the form of Web Proxy Layer which is missing in a three-tier application.

VI. Conclusion and Future Directions

In this paper we have so far presented the design and security analysis of a four-tier web-based telemedicine application. iMedik, the proposed system, differs from existing web-based telemedicine services due to its architecture and its emphasis on application level vulnerabilities. We have illustrated how the introduction of the fourth layer of iMedik or the Web Proxy Layer reduces the risk for many types of attacks. The advantages of the proposed four-tier architecture over existing three-tier architecture have also been highlighted. The security measures incorporated at various layers of the application shows how such a four-tier application can be protected against common web application vulnerabilities. Furthermore, how users can access healthcare data in the proposed system from any location using desktop computers as well as handheld devices has been discussed. The authors firmly believe that the security measures combined with the four-tier architecture presented here gives a simple but elegant road map for secure web application development for anywhere, anytime availability.

The four-tier telemedicine system, as described in Section II, has each layer hosted on a single computer. This scheme works perfectly so long as all the servers are running without failure. However, if any of the servers crashes, the entire system becomes unavailable. To eliminate such faulty situations and to improve server performance, we are presently working towards a high-availability cluster-based implementation of iMedik where each of the layers will form individual clusters. It is to be noted that the authentication module of iMedik having custom routines for user identification, may be replaced by SAML authentication (SIM based authentication for wireless access) in future. Such schemes will help us in performing identity federation of users among multiple hospitals and insurance providers in a flexible and secure manner. We are also integrating a HL7 module into the existing system for ensuring interoperability of our system with other e-Healthcare applications.

Acknowledgment

The authors would like to express their gratitude to the members of the Project Review Committee for their suggestions during the course of the project and to the personnel of WEBEL ECS, Kolkata, India for their help in the deployment of the system. Acknowledgements also go to Dr. Sangeeta D. Bhattacharya, School of Medical Science and Technology, IIT Kharagpur for her continued help and guidance. The contributions of Mr. Debkumar Pradhan, Mr. D. Durga Prasad, Mr. Partieswar Rao, Mr. Abhijit Bhosale and Mr. Suman Kundu during the implementation phase are highly appreciated.

REFERENCES

[1] A.R. Al-Ali, A.O. Abdul Salam, L. Al-Zohlof, M. Manna and R. Zakaria, "A cyber medical center," Computer Methods and Programs in Biomedicine, 80(1), pp. S85–S94, December 2005.

[2] M. Masseroli, A. Visconti, S. G. Bano and F. Pinciroli, "He@lthCo-op: a web-based system to support distributed healthcare co-operative work," Computers in Biology and Medicine, 36(2), pp. 109–127, February 2006.

[3] E. D. Lemaire, D. Deforge, S. Marshall and D. Curran, "A secure web-based approach for accessing transitional health information for people with traumatic brain injury," Computer Methods and Programs in Biomedicine, 81(3), pp. 213–219, 2006.

[4] J. Zhang, J. Sun, Y. Yang, X. Chen, L. Meng and P. Lian, "Web-based electronic patient records for collaborative medical applications," Computerized Medical Imaging and Graphics, 29(2), pp. 115–124, March–April 2005.

[5] C. Caceres, E. J. Gomez, F. Garcia, J. M. Gatell and F. del Pozo, "An integral care telemedicine system for HIV/AIDS patients," International Journal of Medical Informatics, 75(9), pp. 638–642, September 2006.

[6] N. Maglaveras et al., "The citizen health system (CHS): a Modular medical contact center providing quality telemedicine services," IEEE Transactions on Information Technology in Biomedicine 9(3), pp. 353–362, 2005.

[7] M. Wang, C. Lau, F. A. Matsen III and Y. Kim, "Personal health information management system and its application in referral management," IEEE Transactions on Information Technology in Biomedicine, 8(3), 287–297, 2004.

[8] Y. Xiang, Q. Gu and Z. Li, "A distributed framework of web-based telemedicine system," In Proceedings of the 16th IEEE Symposium of Computer-Based Medical Systems 2003, pp. 108–113, 26–27 June 2003.

[9] G. K. Matsopoulos, V. Kouloulias, P. Asvestas, N. A. Mouravliansky, K. K. Delibasis and D. Demetriades, "MITIS: a WWW-based medical system for managing and processing gynecological-obstetrical-radiological data," Computer Methods and Programs in Biomedicine 76(1), pp. 53–71, 2004.

[10] L. Desmet, B. Jacobs, F. Piessens, and W. Joosen, "A generic architecture for web applications to support threat analysis of infrastructural components," In Proc. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK.

[11] "Application architecture: an n-tier approach," http://www.15seconds .com/issue/011023.htm

[12] P. V. Sickel, "Hardware configurations for WebSphere application server production environments," ftp://ftp.software.ibm.com/software/ dw/wes/pdf/0212_vansickel.pdf

[13] A. K. Maji, A. Mukhoty, A. K Majumdar, J. Mukhopadhyay, S. Sural, "Secure healthcare delivery over the web: a multi-tier approach," In Proc. Indian Conference on Medical Informatics and Telemedicine (ICMIT 2006), 18–20 December, 2006, Kharagpur, India.

[14] OWASP, "The ten most critical web application security vulnerabilities, 2007 Update," OWASP Whitepaper, 2007.

[15] J.D. Meier, A. Mackman, S. Vasireddy, M. Dunner, R. Escamilla and A. Murukan, "Improving web application security - threats and counter measures," Microsoft Press, P. 13–43, 2003.

[16] "Web Application Security Consortium: Threat Classification," http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.pdf

[17] Watchfire, "Developing and deploying secure web applications," Watchfire Whitepaper, 2004.

[18] SPI Labs, "Hybrid analysis: an approach to testing web application security," SPI Dynamics Whitepaper, 2006.

[19] D. Gritzalis, C. Lambrinoudakis, D. Lekkas and S. Deftereos, "Technical guidelines for enhancing privacy and data protection in modern electronic medical environments," IEEE Transactions on Information Technology in Biomedicine, 9(3), pp. 413–423, 2005.

[20] "Common Weakness Enumeration (CWE)," http://cwe.mitre.org/

[21] "CAPTCHA," http://en.wikipedia.org/wiki/Captcha

[22] "Medical Privacy - national standards to protect the privacy of personal health information," http://www.hhs.gov/ocr/hipaa/

[23] "HIPAA overview," http://www.nchica.org/hipaaresources/ev/hipaa overview.pdf

[24] "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," Official Journal of the European Communities, No L. 281, pp. 31, November 1995.

[25] "European standards and guidance on privacy and confidentiality in healthcare," http://www.eurosocap.org/eurosocap-standards.aspx

[26] "The Personal Information Protection and Electronic Documents Act," http://www.privcom.gc.ca/legislation/02_06_01_e.asp

[27] "Recommended Guidelines & Standards for Practices of Telemedicine in India," Ministry of Information Technology, Government of India, http://www.mit.gov.in/telemedicine/Report of TWG on Telemed Standardisation.pdf

[28] "Nessus vulnerability scanner," http://www.nessus.org/ download/

[29] "Paros – for web application security assessment," http://www.paros proxy.org/index.shtml