Asian Journal of Information Technology 15 (14): 2457-2463, 2016
ISSN: 1682-3915

Securing Cloud-Based Healthcare Information Systems Using Enhanced Password-Based Authentication Scheme

[1]A. Jesudoss and [2]N.P. Subramaniam
[1]Faculty of Computer Science and Engineering, Sathyabama University, Chennai, Tamil Nadu, India
[2]Department of EEE, Pondicherry Engineering College, Puducherry, India

Abstract: As new and unknown security threats pose a life-threatening issues in medical field, higher levels of security is highly essential to maintain the integrity, confidentiality and authentication of healthcarrecords. Thus this paper presents a unique and an innovative authentication model that protects Cloud-Based Healthcare Information System from various security attacks such as replay attack, password-guessing attack and keylogger attack. Although the primary goal of the work is providing unique and innovative authentication framework, it also satisfies security requirements such as integrity, confidentiality and non-repudiation. The novelty of this approach is to reduce the cost involved in authentication process and also avoid the dependency of external devices for authentication such as mobile OTP, biometric authentication sensors, etc. Through the security analysis, it is clearly shown that the proposed scheme is more secure, efficient and cost effective than the existing authentication schemes. The EPBAS Scheme has been implemented using Java 7.0 and tomcat 7.0.40. The applet client and servlet program has been developed for client-server communication. The AES 256-bit encryption was made with the help of JCE (Java Cryptography Extension). Cryptographic hash function SHA256 (Secure Hash Algorithm) has been implemented. Hence salt bytes must be minimum 32 bytes. The secret key is generated based on the client's password using PBKDF2WithHmacSHA1 (Password-Based key Derivation Function 2 with Hash-based Message Authentication Code Secure Hash Algorithm 1) algorithm.

Key words: Authentication, Attacks, Password, OTP, Replay Attack, Keylogger, Password-Guessing attack

INTRODUCTION

Security is a notable problem in rapidly growing healthcare industries through out the World. Healthcare Information System facilitates the day-to-day work of physicians, surgeons, nurses, Clinicians, administrators and patients as well. The regulations have been devised for providing quality healthcare to the public under Health Insurance Portability and Accountability Act (HIPAA) 1996. Securing medical databases is an important aspect of Healthcare Information System (Pangalos, 1996). Security is the most important aspect of any healthcare applications as they are accessible to general public. Authentication is one of the first and foremost predominant aspects of security. It helps to prevent the unauthorized users from accessing the network, devices, or applications. There are many authentication schemes prevailing in the computing world but each has its advantages and disadvantages as well. The security threats and attacks are increasing day-by-day (Chou, 2013) and it is mandatory to protect our application from such threats and attacks. Particularly, online applications are vulnerable to threats and attacks. Vulnerability assessment is the process of detecting vulnerabilities in the application (Onwubiko and Lenaghan, 2007).

Keylogger attack is an attack using which one can steal the user's password (Raza *et al*., 2012). A Keylogger is a piece of software which records all the keystrokes of the users. In order to prevent this type of attack, financial institutions and banks have introduced the virtual keyboard concept. Virtual Keyboard is a program that resembles a keyboard, where the users have to click the letters to enter the password. Since the actual keyboard is not used, keyloggers will not be able to track the user credentials. However, by capturing the screen of the users, the values entered in the virtual keyboard can be seen. To prevent this, the display of the letters in virtual keyboard is also randomized to avoid computing its coordinates. Hence for every key press, the letters on the keyboard will be shuffled.

Another important attack against authentication is password-guessing attack. Dictionary Attack and Brute-Force Attacks are types of password-guessing attacks. Dictionary Attack is an attack in which the

Corresponding Author: A. Jesudoss, Faculty of Computer Science and Engineering, Sathyabama University, Chennai, Tamil Nadu, India

adversary prepares a list of all known possible words and uses it to crack the password. Brute-Force Attack is an attack which attempts to try all possible combinations for the given set of alphabets, numeric, or special characters. The passwords are hashed and not encrypted for better security (Patel *et al*., 2013). The hashes cannot be reversed and hence it is preferred over the encryption. However, by using brute-force attack, the hashes can also be cracked. But it is very complex and it takes a lot of time and effort to crack the password.

The brute-force attack is simplified by the creation of Rainbow tables (Thing and Ying, 2011). A rainbow table is a table of precomputed hashes for the given length and for the given set of characters. Dictionary Attacks or any attacks against rainbow tables can be prevented by the use of salt. The salt makes it highly difficult to find the password. The time taken to validate the credentials is another important factor to be considered.

Replay attack is a type of man-in-the-middle attack in which the adversary captures the packet, manipulates the information present in the packet and transmits it to the victim (Syverson *et al*., 1994). One-Time Password (OTP) helps to avoid replay attacks. Mobile based OTPs are becoming popular these days (Mijin Kim *et al*., 2009). By spoofing the SIM card, it is quite possible to hack the OTP sent to the mobile. Nowadays, all the Web applications support multifactor authentication such as mobile OTP, email OTP, etc.

Literature review: The security model proposed in dynamic password authentication scheme may not be vulnerable to replay attack as the credentials password and time stamp are hashed but it is vulnerable to keylogger attack. Moreover, the hashes are not encrypted and therefore it is vulnerable to brute-force attack through the use of rainbow tables. Therefore in the proposed work, the hashed credentials are further encrypted. The scheme proposed by Eman *et al*. (2011) shows that Kerberos protocol is vulnerable to password-guessing attacks and the encryption algorithm used is Triple-DES (Data Encryption Standard). AES (Advanced Encryption Standard) algorithm is better than Triple-DES as in the case of the latter some issues may arise if data encrypted is >32 gb with single key, whereas the AES has higher limit (http://security.stackexchange.com/).

The Enhanced Kerberos Authentication Scheme shows that the application can be protected against password-guessing attack but it is vulnerable to keylogger attack (Jesudoss and Subramaniam, 2014). Since the formula used in this approach is computed automatically, if the adversary steals the password using a keylogger, there will be nothing to stop him from accessing the application. Of course, it may protect against replay attack but not against keylogger attack. Vulnerability in authentication process may arise due to the lack of knowledge on the part of the end user or due to social engineering attack or due to insufficient security to the system. Vulnerability also arises during the transmission or may arise due to the lack of security on the server side (Renaud, 2004). For making secured transaction, HTTPS (Hypertext Transfer Protocol Secure) may be used. As HTTP (Hypertext Transfer Protocol) protocol is a stateless protocol and is not encrypted, the adversary can manipulate the HTTP request. In order to rectify this, SSL (Secure Socket Layer) may be used or other means of encryption may be implemented (Miculan and Urban, 2011). Secure coding is one main aspect that helps to avoid vulnerabilities. In fact, choosing the programming language contributes to the successful development of flawless secured application. Java is considered to be more secured than C and C++ as it does not support low-level features. Moreover, Java is platform-independent, portable and distributed in nature.

The fingerprint biometric authentication may provide additional layer of security to the applications. It is considered to be secured as the fingerprints are uniquely identified, difficult to forge and need not be remembered. On the contrary, it is also vulnerable as it may be faked or it may be intercepted based on the time available, efforts made and resources available, etc (www.digitalpersona.com). The credentials of the user may be stolen either offline or online. Offline attacks may include malwares, phishing, lack in antivirus, firewall, browser, software or operating system updates, etc. Online attacks may include session hijacking, masquerading or eavesdropping on the network (Hiltgen *et al*., 2006). Chaos-based encryption technique shows how images can be used for encryption and protects against attacks using permutation-diffusion architecture (Wang *et al*., 2014). Chaotic Image-based encryption with DNA coding can also guard against a variety of security attacks (Zhang and Liu, 2013).

Single Sign-on is an authentication technique that allows a user to enter the username and password once and be able to log on to multiple websites without any credentials. Some examples of SSO (Single Sign-On) protocols are Liberty Alliance, SAML (Security Assertion Markup Language), Shibboleth, OpenID, WS (Web Service) Federation, etc. The SSO like SAML and OpenID are vulnerable to masquerading attack. It allows the malicious user to impersonate and access the resource (Armando *et al*., 2013). Security risks are very higher for SSO protocols. Therefore, multifactor authentication must be used for such cases. The pomcor. com explains the need for extra passwords, particularly in mobile devices (pomcor.com). The usability is the most important point to be kept in mind while adding more layers of security. When security is increased, the usability of the application should not be affected. Both should be balanced equally (Mathew and Thomas, 2013).

Motivation: Many security authentication schemes are available to protect the application against different types of attacks. For example, Kerberos is considered to be the best among the authentication protocols but it is prone to replay and password guessing attacks. Digital certificates are the good source of proving one's identity. But they are very expensive and require knowledge on the part of the end users to implement, maintain and update the certificates.

Therefore, it has become necessary to depend on an additional security mechanism. An Authentication scheme which is good at one side may be lacking in many security features at the other side. It is a fact that there is no authentication scheme that is good and perfect in all aspects. It may offer protection against some attacks but remains vulnerable to many attacks. But, there may be some features which are commonly missing in many authentication schemes. It is essential to protect applications against new possible attacks by incorporating all security features available. In the proposed work, a security model has been developed to provide security against three types of attacks, namely replay security issues through a single security authentication model.

MATERIALA AND METHODS

Novel EPBAS scheme: The architecture of the proposed work is illustrated in the Fig. 1. The proposed authentication scheme consists of three phases. They are Phase. The notations used throughout this paper are given below. In Fig. 1, the Login Phase is demonstrated from steps 1-16 and the Authentication Phase is demonstrated from steps 17-27.

The steps for implementation are given below:

- Step 1: After time synchronization using NTP (Network Time Protocol), the client makes an initial request to the server by providing credentials
- Step 2: The server checks whether the user is a legitimate one or not by verifying the credentials on the database
- Step 3: The data pertaining to the user are retrieved for verification purpose
- Step 4: The server initiates the process for generation of nonce, salt and a new session key
- Step 5: The process returns the result of the nonce value, salt value and session key
- Step 6: The server stores the nonce value, salt value, session key and the timestamp on the database against the username
- Step 7: The server passes these four values for encryption
- Step 8: Encrypted result is returned to the server
- Step 9: The server sends the encrypted response to the client
- Step 10: The client passes encrypted response for decryption
- Step 11: The decrypted result is returned to the client
- Step 12: The client initiates the process for the computing of final nonce value (i.e., secret value) and generates polymorphic password
- Step 13: Results are returned to the client, i.e., computed final nonce value, computed hash value (i.e., password+salt+timestamp) and time stamp
- Step 14: The client passes these results for encryption which is carried out by new session key

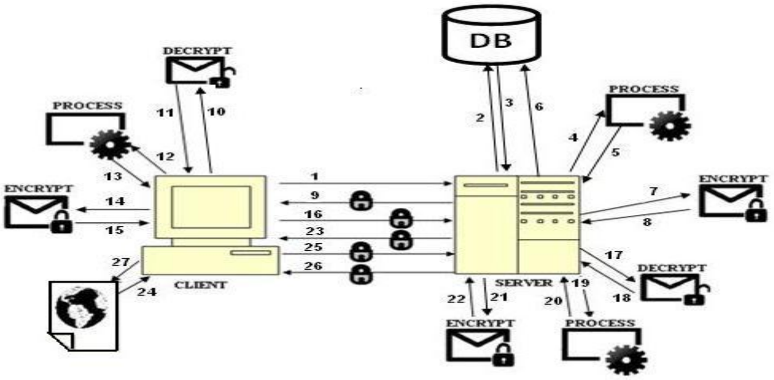


Fig. 1: Architecture of authentication model for secured health care information systems

- Step 15: The encrypted results are returned to the client
- Step 16: The client sends the encrypted results to the server
- Step 17: The server passes the encrypted results to request for decryption
- Step 18: The decrypted results are returned to the server
- Step 19: The server initiates the verification process such as the validity of the client, correctness of the polymorphic password and the final nonce value
- Step 20: The results are returned to the server. If the results are positive, the server sends the service token to the client
- Step 21: The service token credentials are passed for encryption which is carried out by the new session key and the values are cleared from the database
- Step 22: The service Token is returned to the server after encryption
- Step 23: The service token is sent to the client
- Step 24: The client enters the data for accessing a particular service
- Step 25: The client makes the request for the service to the server using service token
- Step 26: The service is granted by the server
- Step 27: The client consumes the service

The application used in the proposed work is Cloud-based healthcare application. The underlying application can be changed according to our requirements. This security model will be very much useful for applications where security is of utmost importance such as banking transactions, share trading portals or any application where monetary transaction is involved such as financial companies or where top secret is to be maintained such as defence applications.

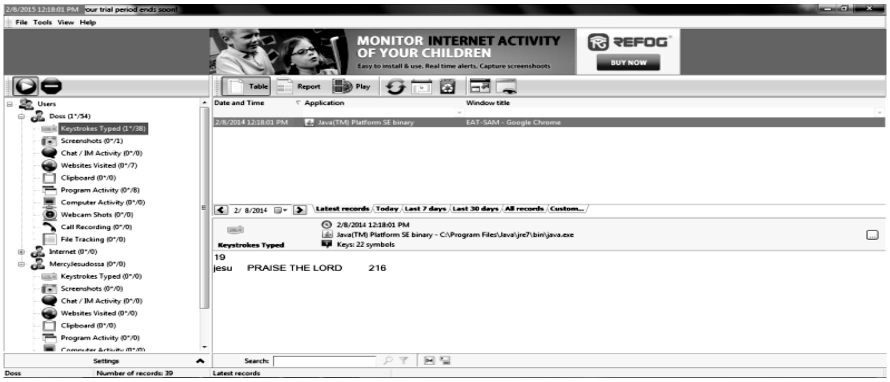
Security analysis of EPBAS: The EPBAS scheme provides an enhanced attack-tolerant security authentication model by protecting against three security attacks, namely replay attack, password-guessing attack and keylogger attack. In the proposed system, keylogger attack and screen-capture attack are handled much more dynamically. In the existing system, a particular security authentication scheme may defend one either against keylogger attack or screen-capture attack but not against both. But the proposed system helps to shield from many attacks including keylogger and screen-capture attacks. Keylogger attack will not be effective in the proposed authentication model due to the fact that the password is not the only requirement to access the web application.

The password PWi and the computed nonce value i.e., N2i are both required to fulfill the verification process. The secret shared is not just these two parameters but there is also a third parameter called Secret Value i.e., SVi which is used for computing Computed Nonce Value, i.e., N2i.

RESULTS AND DISCUSSION

Figure 2 shows that the Refog is not able to track the complete credentials of the user. It can track only "username", "password" and "final nonce value". These credentials are useless without knowing the salt or the secret value (i.e., SVi). Moreover, the final nonce value is a One-Time Password (OTP). It also shows that the Keylogger can access the password entered on the user's password text box but it cannot use it with out the other two secret parameters, N2i and SVi. Since, the N2i value is unique for every transaction, the keylogger attack can be completely avoided by the proposed work.

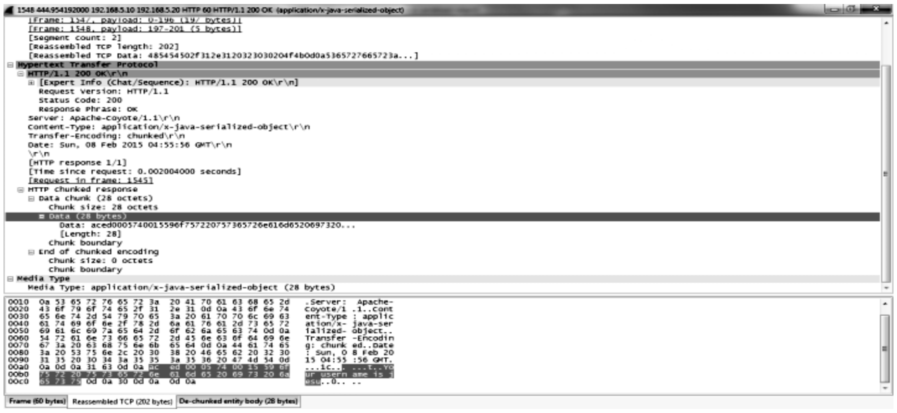
Figure 3 shows the packet captured during the initial request from the client. The username is shown in the output. Figure 4 shows the packet being captured when the client submits the encrypted credentials to the server. The output displayed is encrypted using AES256 and is unbreakable. The Healthcare Information System Login Screen is shown in Fig. 5 (Table 1-4).



```
2/8/2014 12:18:01 PM   Java(TM) Platform SE binary   EAT-SAM - Google Chrome

Keystrokes Typed   2/8/2014 12:18:01 PM
                   Java(TM) Platform SE binary - C:\Program Files\Java\jre7\bin\java.exe
                   Keys: 22 symbols
19
jesu    PRAISE THE LORD       216
```

Fig. 2: Refog output

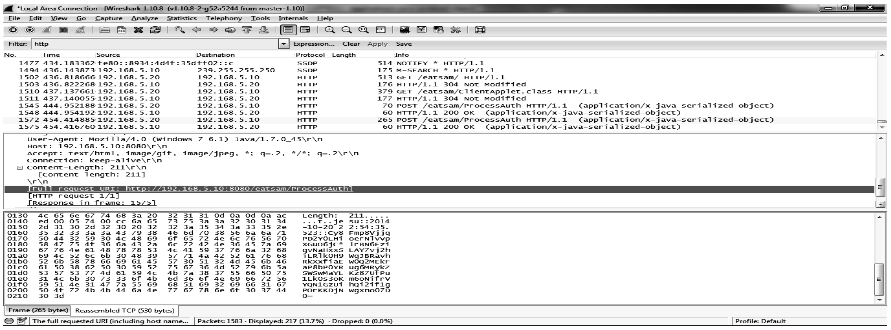


```
1548 444.954192000 192.168.5.10 192.168.5.20 HTTP 60 HTTP/1.1 200 OK (application/x-java-serialized-object)
    [Frame: 1547, payload: 0-196 (197 bytes)]
    [Frame: 1548, payload: 197-201 (5 bytes)]
    [Segment count: 2]
    [Reassembled TCP length: 202]
    [Reassembled TCP Data: 485454502f312e3120323030204f4b0d0a5365727665723a...]
⊟ Hypertext Transfer Protocol
  ⊟ HTTP/1.1 200 OK\r\n
    ⊞ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Request Version: HTTP/1.1
      Status Code: 200
      Response Phrase: OK
    Server: Apache-Coyote/1.1\r\n
    Content-Type: application/x-java-serialized-object\r\n
    Transfer-Encoding: chunked\r\n
    Date: Sun, 08 Feb 2015 04:55:56 GMT\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.002004000 seconds]
    [Request in frame: 1545]
  ⊟ HTTP chunked response
    ⊟ Data chunk (28 octets)
        Chunk size: 28 octets
      ⊟ Data (28 bytes)
          Data: aced0005740015596f757220757365726e616d6520697320...
          [Length: 28]
        Chunk boundary
    ⊟ End of chunked encoding
        Chunk size: 0 octets
        Chunk boundary
⊟ Media Type
    Media Type: application/x-java-serialized-object (28 bytes)

0010  0a 53 65 72 76 65 72 3a  20 41 70 61 63 68 65 2d   .Server:  Apache-
0020  43 6f 79 6f 74 65 2f 31  2e 31 0d 0a 43 6f 6e 74   Coyote/1 .1..Cont
0030  65 6e 74 2d 54 79 70 65  3a 20 61 70 70 6c 69 63   ent-Type : applic
0040  61 74 69 6f 6e 2f 78 2d  6a 61 76 61 2d 73 65 72   ation/x- java-ser
0050  69 61 6c 69 7a 65 64 2d  6f 62 6a 65 63 74 0d 0a   ialized- object..
0060  54 72 61 6e 73 66 65 72  2d 45 6e 63 6f 64 69 6e   Transfer -Encodin
0070  67 3a 20 63 68 75 6e 6b  65 64 0d 0a 44 61 74 65   g: chunk ed..Date
0080  3a 20 53 75 6e 2c 20 30  38 20 46 65 62 20 32 30   : Sun, 0 8 Feb 20
0090  31 35 20 30 34 3a 35 35  3a 35 36 20 47 4d 54 0d   15 04:55 :56 GMT.
00a0  0a 0d 0a 31 63 0d 0a ac  ed 00 05 74 00 15 59 6f   ...1c... ...t..Yo
00b0  75 72 20 75 73 65 72 6e  61 6d 65 20 69 73 20 6a   ur usern ame is j
00c0  65 73 75 0d 0a 30 0d 0a  0d 0a                     esu..0.. ..
```

Frame (60 bytes)
Reassembled TCP (202 bytes)
De-chunked entity body (28 bytes)

Fig. 3: Initial client request captured



Filter: http

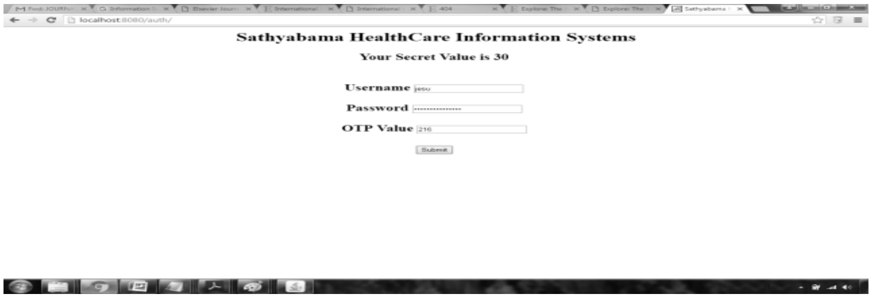
```
No.   Time       Source                    Destination      Protocol Length Info
 1477 434.183362 fe80::8934:4d4f:35d ff02::c          SSDP          514 NOTIFY * HTTP/1.1
 1494 436.143873 192.168.5.10        239.255.255.250  SSDP          175 M-SEARCH * HTTP/1.1
 1502 436.818666 192.168.5.20        192.168.5.10     HTTP          513 GET /eatsam/ HTTP/1.1
 1503 436.822268 192.168.5.10        192.168.5.20     HTTP          176 HTTP/1.1 304 Not Modified
 1510 437.137661 192.168.5.20        192.168.5.10     HTTP          379 GET /eatsam/ClientApplet.class HTTP/1.1
 1511 437.140055 192.168.5.10        192.168.5.20     HTTP          177 HTTP/1.1 304 Not Modified
 1545 444.952188 192.168.5.20        192.168.5.10     HTTP           70 POST /eatsam/ProcessAuth HTTP/1.1  (application/x-java-serialized-object)
 1548 444.954192 192.168.5.10        192.168.5.20     HTTP           60 HTTP/1.1 200 OK  (application/x-java-serialized-object)
 1572 454.414885 192.168.5.20        192.168.5.10     HTTP          265 POST /eatsam/ProcessAuth HTTP/1.1  (application/x-java-serialized-object)
 1575 454.416760 192.168.5.10        192.168.5.20     HTTP           60 HTTP/1.1 200 OK  (application/x-java-serialized-object)

    User-Agent: Mozilla/4.0 (Windows 7 6.1) Java/1.7.0_45\r\n
    Host: 192.168.5.10:8080\r\n
    Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2\r\n
    Connection: keep-alive\r\n
  ⊟ Content-Length: 211\r\n
      [Content length: 211]
    \r\n
    [Full request URI: http://192.168.5.10:8080/eatsam/ProcessAuth]
    [HTTP request 1/1]
    [Response in frame: 1575]

0130  4c 65 6e 67 74 68 3a 20  32 31 31 0d 0a 0d 0a ac   Length:  211.....
0140  ed 00 05 74 00 cc 6a 65  73 75 3a 3a 32 30 31 34   ...t..je su::2014
0150  2d 31 30 2d 32 30 20 32  32 3a 35 34 3a 33 35 2e   -10-20 2 2:54:35.
0160  35 32 33 3a 3a 43 79 38  46 6d 70 38 56 6a 6a 71   523::Cy8 Fmp8Vjjq
0170  50 44 32 59 30 4c 48 69  6f 65 72 4e 6c 76 56 70   PD2Y0LHi oerNlvVp
0180  58 47 75 4f 36 6a 43 2a  6c 72 42 4e 36 45 7a 69   XGuO6jC* lrBN6Ezi
0190  67 76 4e 61 48 78 78 53  4c 41 59 37 76 6a 32 68   gvNaHxxS LAY7vj2h
01a0  69 4c 52 6c 6b 30 48 39  57 71 4a 42 52 61 76 68   iLRlk0H9 WqJBRavh
01b0  52 6b 58 78 66 69 61 45  57 30 51 32 4d 45 6b 46   RkXxfiaE W0Q2MEkF
01c0  61 50 38 62 50 30 59 52  75 67 36 4d 52 79 6b 5a   aP8bP0YR ug6MRykZ
01d0  53 57 53 77 4d 61 59 4c  4b 7a 38 37 55 66 50 75   SWSwMaYL Kz87UfPu
01e0  31 4c 6b 30 73 33 6f 4b  6d 36 6f 4e 69 66 72 56   1Lk0s3oK m6oNifrV
01f0  59 51 4e 31 47 7a 55 69  68 51 69 32 69 66 31 67   YQN1GzUi hQi2if1g
0200  50 4f 72 4b 4b 44 6a 4e  77 67 78 6e 6f 30 37 44   POrKKDjN wgxno07D
0210  30 3d                                              0=
```

The full requested URI (including host name... | Packets: 1583 · Displayed: 217 (13.7%) · Dropped: 0 (0.0%) | Profile: Default

Fig. 4: Client credentials submission captured



Sathyabama HealthCare Information Systems

Your Secret Value is 30

Username jesu

Password ••••••••••••••••

OTP Value 216

Fig. 5: HealthCare information system login screen

Table 1: Notations used in this paper

Notation	Description
C_i	ith client
S	Server
Un_i	Username of the ith client
Pw_i	Password of the ith client
Na_i	Network Address of the ith client
$T0_i$	Initial Timestamp of the ith client
ISK_i	Initial Session Key
$N1_i$	Nonce Value
$S1_i$	Salt Value generated by the server
TSK_i	Temporary Session Key
$T1_i$	Timestamp of Server
Acknowledgment	
$N2_i$	Computed Nonce value
Sv_i	Secret Value held by the user
Pp_i	Polymorphic Password
$T2_i$	Timestamp of client submission
$?T_i$	Maximum Transmission Delay
h(.)	One-way hash function
A || B	A and B are Concatenated
E(,)	Encrypt
D(,)	Decrypt
St_i	Service Token for ith user

Table 2: Registration Phase

Client (Ci)	Server (S)
Ci chooses Ui and PWi	
Ci fetches Nai automatically	
Sends {Ui, Pwi and Nai} securely	
	Stores { Ui, Pwi, Nai } on Table
	Generates ISKi
	Sends E(ISKi, Pwi)

Table 3: Login phase

Client (Ci)	Server (S)
Time synchronized using NTP	
Ci enters Ui	
Sends {Ui, E(Ui || Nai || T0i, ISKi)}	
	Searches Ui in Table
	Finds appropriate ISKi
	D(Ui || NAi || T0i, ISKi)
	Verifies NAi
	Computes ?Ti
	Generates TSKi and N1i
	where 1 N1<100
	And N1 mod 5 = 0
	Generates S1i
	Computes T1iStores {TSKi, N1i, S1i, T1i}Sends
{E(TSKi || N1i || S1i || T1i, ISKi)}	
D(TSKi || N1i || S1i || T1i, ISKi)	
Computes N2i = (N1i / SVi)3	
Computes T2i and	
PPi = h(PWi || S1i || T1i)	
Sends	
{E(Ui ||T2i || N2i || PPi, TSKi)}	

Table 4: Authentication phase

Client (Ci)	Server (S)
	Searches Ui in Table
	Finds appropriate TSKi
	E(Ui ||T2i || N2i || PPi, TSKi)
	Computes ?Ti
	Computes PPi' = h(PWi || S1i || T1i)
	Compares PPi == PPi'
	Sends { E(STi, TSKi) } from Table
	Clears S1i, N1i, TSKi
D(STi, TSKi)	
Synchronized	Synchronized

CONCLUSION

The EPBAS Scheme proves the mutual authentication between the server and the client by means of encrypted message using secret key from the server and encrypted message using the new session key by the client and hence protects the healthcare information system from the malicious adversaries. The proposed system does not allow to masquerade or alter the message, to view the message, or to deny the message that is sent. Therefore, it satisfies various requirements of security such as authentication, integrity, confidentiality and non-repudiation. A good authentication system is the one which has satisfied the three basic security requirements, namely "what you know", "what you have" and "what you are". The password entered by the users satisfies the "what you know" parameter. The salt and secret key satisfies the "what you have" parameter. It is usually the biometric marks of the user that serves as "what you are" parameter" but in the proposed method, the secret value i.e., SVi is used as "what you are" parameter. Usually the biometric marks entered by the user identifies the User Id using some image processing algorithm and it hashes, encrypts and sends it without storing anywhere. Similarly the proposed method uses secret value (i.e., SVi) which is not stored anywhere in the client or the server or anywhere else. This feature makes the secret value (i.e., SVi) to prove the identity of the user uniquely. Hence, it is considered as "who you are" parameter. It provides Multi-Factor Authentication, endpoint to endpoint security and unique OTP for every transaction. Hence it protects against various security attacks such as keyloggers, replay attack and password-guessing attack. Therefore, the Proposed System not only gives protection from security attacks but also it reduces the cost involved in the authentication processs and avoids the usage of external devices for authentication. Therefore it proves that it has all the essential features of a good authentication system.

RECOMMENDATIONS

As a future scope of this research, the fingerprint biometric authentication may be implemented. GPS-based authentication will be more effective in authenticating the Web users.

REFERENCES

Armando, A., R. Carbone, L. Compagna, J. Cuellar, G. Pellegrino and A. Sorniotti, 2013. An authentication flaw in browser-based Single Sign-On protocols: Impact and remediations. Comput. Security, 33: 41-58.

Chou, T.S., 2013. Security threats on cloud computing vulnerabilities. Int. J. Comput. Sci. Inf. Technol., 5: 79-88.

Emam, E.El., M. Koutb, H.M. Kelash and O.S. Faragallah, 2011. An authentication protocol based on kerberos 5. Int. J. Network Secur., 12: 159-170.

Hiltgen, A., T. Kramp and T. Weigold, 2006. Secure internet banking authentication. IEEE. Secur. Privacy, 4: 21-29.

Jesudoss, A. and N.P. Subramaniam, 2014. Enhanced kerberos authentication for distributed environment. J. Theor. Appl. Inf. Technol., 69: 368-374.

Kim, M., B. Lee, S. Kim and D. Won, 2009. Weaknesses and improvements of a one-time password authentication scheme. Int. J. Future Generation Commun. Networking, 2: 29-38.

Mathew, G. and S. Thomas, 2013. A novel multifactor authentication system ensuring usability and security. Preprint, 2: 21-30.

Miculan, M. and C. Urban, 2011. Formal analysis of facebook connect single sign-on authentication protocol. SOFSEM., 11: 22-28.

Onwubiko, C. and A.P. Lenaghan, 2007. Managing security threats and vulnerabilities for small to medium enterprises. Proceedings of the IEEE Conference on Intelligence and Security Informatics, May 23-24, 2007, New Jersey, pp: 244-249.

Pangalos, G.J., 1996. Secure medical databases: Design and operation. Int. J. Biomed. Comput., 43: 53-60.

Patel, P.N., J.K. Patel and P.V. Virparia, 2013. A cryptography application using salt hash technique. Int. J. Appl. Innovation Eng. Manage. IJAIEM., 2: 236-239.

Raza, M., M. Iqbal, M. Sharif and W. Haider, 2012. A survey of password attacks and comparative analysis on methods for secure authentication. World Appl. Sci. J., 19: 439-444.

Renaud, K., 2004. Quantifying the quality of web authentication mechanisms: A usability perspective. J. Web Eng., 3: 95-123.

Syverson, P., 1994. A taxonomy of replay attacks. Proceedings of the 7th IEEE Workshop on Computer Security Foundations CSFW, June 14-16, 1994, IEEE, Franconia, Hampshire ISBN: 0-8186-6230-1, pp: 187-191.

Thing, V.L. and H.M. Ying, 2011. Rainbow table optimization for password recovery. Int. J. Adv. Software, 4: 479-488.

Wang, B., X. Wei and Q. Zhang, 2014. A novel and fast chaotic cryptosystem for image encryption. J. Comput. Theor. Nanosc., 11: 731-738.

Zhang, Q. and L. Liu, 2013. DNA coding and chaos-based image encryption algorithm. J. Comput. Theor. Nanosci., 10: 341-346.