2009 Pacific-Asia Conference on Circuits,Communications and System

A Scheme to Improve Security of SSL

Zhao Huawei[1,2] and Liu Ruixia[2]
[1]School of Computer and Information Engineering
Shandong University of Finance
Jinan, China
[2]Shandong Computer Science Center
Jinan, China
e-mail:zhuav@163.com

***Abstract*—SSL is a cryptographic protocol that is widely used in secure applications based on web browser, such as e-payment. The protocol is based on the principle of PKI and uses digital certificates to realize secure communication. However, the use of certificate in SSL does not obey the strict hierarchy CAs, which causes some secure defects in SSL. Besides, session key of SSL is limited by the exportation of USA, and its valid length is only 40 bits, and the length is vulnerable to exhaustive attack. A scheme proposed to improve security of SSL by modifying hand-shake protocol. Analyzing the scheme shows that it not only can remedy the secure defects of SSL, but also can add the valid length of session key in SSL.**

Keywords- SSL;Session key; Exhaustive attack

I. INTRODUCTION

SSL[1-3] is a cryptographic protocol that is based on the principle of PKI[4], and uses digital certificate to provide secure services such as confidentiality, data integrity and identity authentication for communications over Internet[5]. Because now many applications are based on the structure of Browser/Server, SSL is in wide-spread use in these applications, and provides security for them using HTTPS protocol[6].

However, SSL is not a robust secure protocol in the real world. The first reason is that SSL does not obey the strict trust model of PKI: The principle of PKI is very complicated, and needs a strict trust model to ensure security. However, the cost of maintaining a strict trust model is too high to be accepted by the public. So in order to be used widely, SSL has to make the tradeoff between usability and security, and adopts a non-strict trust model: web model, which could cause an adversary to use crafty method to attack SSL. The second reason is the valid length of SSL session key is too short to resist exhaustive attack: SSL is a secure protocol developed by Netscape that is a USA company, however, due to the exportation limitation of USA, the valid length of session key in SSL is only 40 bits, and in fact the length is vulnerable to exhaustive attack.

A scheme is given in the paper to strengthen the security of SSL. The scheme modifies the flow of hand-shake protocol and changes the method of generating session key. Analysis shows the scheme can resist crafty attack and exhaustive attack. In this paper, section two gives the structure of SSL, section three shows security defects of SSL, section four proposes a scheme to improve security of SSL and a conclusion is given in section five.

II. STRUCTURE OF SSL

SSL consists of two sub-protocols. One is hand-shake protocol, the other is record protocol. Hand-shake protocol is the key of SSL, and it realizes certificates exchanging, key materials exchanging and identity authentication. The function of record protocol is using the session keys produced in hand-shake protocol to encapsulate the data to be exchanged. And the encapsulation can provide confidentiality and integrity for data. Because we will modify hand-shake protocol in the remainder section, here, we give the illustration of a full hand-shake protocol[7].

1, A client sends a ClientHello message to a server to ask for a SSL connection, and the message specifies the highest version SSL version it supports, a random number, a list of suggested cipher suits and compressed method.

2, The Server responds with a ServerHello message, containing the chosen protocol version, a random number, cipher suite, and compression method from the choices offered by the client. The server may also send a session id as part of the message to perform a resumed handshake.

3, The Server sends its Certificate message (depending on the selected cipher suite, this may be omitted by the Server).

4, The Server sends a ServerHelloDone message, indicating it is done with handshake negotiation.

5, The Client responds with a ClientKeyExchange message, which may contain a Pre-Master Secret, public key, or nothing. (Again, this depends on the selected cipher.)

6, The Client and Server use the random numbers and Pre-Master Secret to compute a common secret, called the "master secret", and then the both parts use the master secret to compute a key-block. All other key data for this connection is derived from this key-block (and the client- and server-generated random numbers), which is passed through a carefully designed "pseudorandom function". And the key data includes two session keys: client_write_key and server_write_key.

7, The Client now sends a ChangeCipherSpec record, essentially telling the Server, "Everything I tell you from now on will be encrypted." Note that the ChangeCipherSpec is itself a record-level protocol.

National Natural Science Foundation of China(No:60802030);
National Natural Science Foundation of China(No:60743006);
Shandong Province Natural Science Foundation (No:Y2007G15);
Foundation of Shandong Provincial Education Department (No:J07YJ05);
Doctor Foundation of Shandong University of Finance(No:06BSJJ07)

978-0-7695-3614-9/09 $25.00 © 2009 IEEE
DOI 10.1109/PACCS.2009.148

8, Finally, the Client generates a Finished message containing a hash and MAC over the previous handshake messages, and encrypts it using the client_write_key before sending it.

9, The Server will attempt to decrypt the Client's Finished message by the client_write_key, and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.

10, Finally, the Server sends a ChangeCipherSpec and its encrypted Finished message, and the Client performs the same decryption and verification.

At this point, the "handshake" is complete and the application protocol is enabled. Application messages exchanged between Client and Server will be encrypted.

III. SECURITY ANALYSIS OF SSL

SSL is in wide-use spread in web browser applications, but it is not secure enough. In this section, we first point out the root of insecurity and then give two secure defects of SSL.

A. Root of insecurity

The security of SSL is based on PKI, and before the deployment of PKI, one of four primary trust models could be chosen: strict hierarchy of CAs, distributed trust architecture, web model, and user-centric model. The four trust models have advances respectively, and web model is suitable to the large-scale PKI. Thus, in order to be used widely, SSL based on web browser adopts web model.

In web model, there are two trust roots. One is root certificates installed in web browser; the other is user's judgment. When an user wants to establish a SSL connection with a server, the server will send out his server certificate and let web browser of the user uses the public key in it to encrypt pre-master key. Once web browser receives the server certificate, it will check validity of the certificate firstly. If the superior certificate (it is a root certificate commonly) that issues the server certificate is in the user's web browser, validity of the server certification will be verified automatically, and here the superior certificate is a kind of trust root. In other case, if there is no superior certificate in web browser, then web browser cannot verify the server certificate, now, it will pop up a message dialog to tell the user to decide whether to trust the server certificate or not, and here the user's judgment is the other kind of trust root.

From above, we can see that the verification of server certificate is not strict, which causes two security defects.

B. Attack from evil root certificate

Root certificate is a special kind of certificate, and commonly, it is self-signed. There are two origins of root certificate in web browser: pre-installation and user installation. And in the latter origin, the root certificate is not verified strictly. For example, if a user finds a certificate is a self-signed root certificate, and the issuer information in it is right, then he might trust it and install it in web browser. Thus, some evil root certificates faked by an adversary could be installed by this way. For example, an adversary could generate a self-signed root certificate, and its issuer information shows it is a trusted CA, however its public key belongs to the adversary. Now, when a user installs the faked root certificate in his web browser and uses it to encrypt a pre-master key, the adversary can get the pre-master key using his private key according to the public key. And what more, if the adversary uses the private key according to the evil root certificate to sign some server certificate, these certificates could be trusted by the user's web browser automatically. And it would goes to further security threaten.

C. Short key length

In hand-shake protocol, client and server both will compute two session keys (client-write-key and server-write-key) with a key-block, client- and server-generated random numbers. However, due to the limitation of exportation of USA, the length of key-block is only 42 bytes long，of which the 32^{nd} to 36^{th} bytes are used to generate client-write-key; the 37^{th} to 41^{st} bytes are used to generate server-write-key.

Client-write-key:

$$MD5(Key-Block_{32...36} \,\|\, randomc \,\|\, randoms)_{0...15} \tag{1}$$

Server-write-key:

$$MD5(Key-Block_{37...41} \,\|\, randomc \,\|\, randoms)_{0...15} \tag{2}$$

Here randomc and randoms are client- and server-generated random numbers respectively.

Because the both random numbers are public in hand-shake protocol, only 5 bytes of key-block keeps secret to adversary. In other words, the valid lengths of both session keys are only 40 bits. In fact, a key of 40 bits is vulnerable to exhaustive attack.

IV. A SCHEME TO STRENGTHEN SECURITY

In this section, we give a scheme to remedy the security defects mentioned above, and it is fit for secure business based on web browser, such as e-payment.

A. Modified hand-shake protocol

From discussion above, we know that a root certificate installed in web browser maybe a faked evil certificate, which causes there is no credible data to establish SSL connection. Now, we improve SSL protocol by modifying the flow of hand-shake and adding a credible data in it:

First, before a user wants to establishes a SSL connection, he should send a request to a server by dialing a customer number using a mobile phone, and when the server receives the request, he should generate and return a random number as a deal password by a short message, and at the same time, saves the deal password and the user's phone number into its database. The process is shown in Fig.1.

Next, the user should use his web browser to visit the Home Site of the server, and then hand-shake protocol in SSL begins to run.

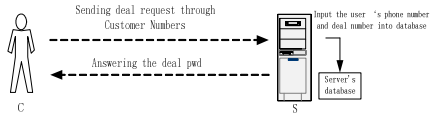

Figure 1. Deal password exchanging

The modified hand-shake protocol is shown in Fig.2. In the 1st step of hand-shake protocol, we add the user' phone number into the ClientHello message, and the server will use it later to look for the corresponding deal password. In the 6th step (after the web browser receives the certificate of the server, and before it computes master key), we pop up a dialog to let the user input the deal password. And then web browser will use the deal password, two random number and other key materials to compute master key, key-block and session keys. Symmetrically, the server will perform the same operations to compute keys.

B. Generating keys

Below, we show our solution to strengthen the security of keys in SSL.

Here, we use the deal password as a credible secret value to transform the two public random values to cause the transformed values secret to an adversary. Then we use the transformed random numbers to compute keys.

In our scheme, we use the idea of Hill cipher[8] to transform random numbers:

Suppose part of the 32-byte random number of the user is:

10101100 10111010 10101011
01001010 10101001 01111111
…

We look each byte of the random values as a block.

Suppose the deal password is an 8-byte character string: 1s3df4rc. And we decode the deal password into ASCII codes:

00110001 01110011 00110011 01100100
01100110 00110100 01110010 01100011

We use this decoded 64 bits data as a permutation to transform each random block. The transformation of the first block is shown in Fig.3, and the result is 11000011.

The second block is transformed in the same way, and the result is 01110011.

Following above performance, we can get a new 64-byte secret data to generate keys in SSL.

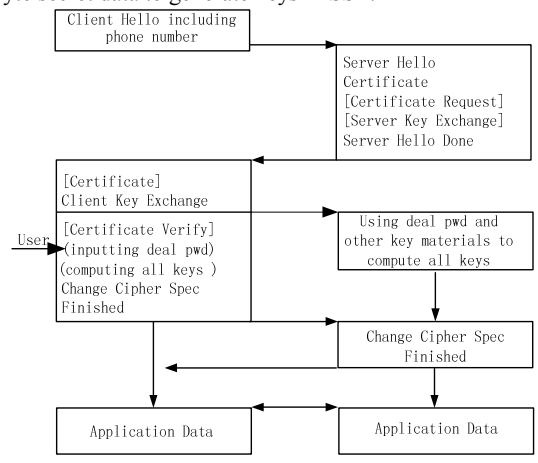

Figure 2. Modified hand-shake protocol

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \bmod 2$$

Figure 3. Transformation of each random block

For example:

The master key is:

$MD5(pre\_master\_secret \parallel SHA(A \parallel pre\_master\_secret \parallel Transformed\ random)) \parallel$
$MD5(pre\_master\_secret \parallel SHA(BB \parallel pre\_master\_secret \parallel Transformed\ random)) \parallel$
$MD5(pre\_master\_secret \parallel SHA(CCC \parallel pre\_master\_secret \parallel Transformed\ random))$

The key-block is:

$MD5(master\ secret \parallel SHA(A \parallel master\ secret \parallel Transformed\ random)) \parallel$
$MD5(master\ secret \parallel SHA(BB \parallel master\ secret \parallel Transformed\ random)) \parallel$
$MD5(master\ secret \parallel SHA(CCC \parallel master\ secret \parallel Transformed\ random))$

The session key of the user is:

$MD5\{key\_block_{32\ldots36} \parallel Transformed\ random\}$

In addition, the transformed random values are:

$f(randomc_0) \parallel \ldots \parallel f(randomc_{32}) \parallel f(randoms_0) \parallel \ldots \parallel f(randoms_{32})$

And $f()$ is a permutation function mentioned above.

C. Security analysis

Now, we make a security analysis for our scheme, and show the scheme can resist the security defects mentioned in section III.

1) Resisting faked evil root certificate

In our scheme, when the adversary installs a faked evil root certificate into the user's web browser, he can get the encrypted per-master key using his private key, but he can not compute the correct keys (including master key, key-block, two session keys and MAC keys). The reason is: after the user sends encrypted pre-master key, he will use the secret deal password to transform the public random numbers, and use the transformed numbers to compute all kinds of keys. Due to without the deal password, an adversary can not compute the correct session keys and MAC keys. So the adversary cannot compute the right Server's Finished message to cheat the user.

2) Resisting exhaustive attack

In the scheme, we make use of the idea of Hill cipher to encrypt public random numbers. Because the encrypted random numbers is the input of MD5 function, the adversary cannot get them from network as clues to achieve the deal password and then to achieve the correct keys. So, if the adversary wants to compute the correct session keys, the only thing he could do is make exhaustive attack. Because in the structure of the "Finished" message, the first four bytes are 14 00 00 24, the adversary can use the four bytes as the proof to guess the right session key in his exhaustive attack.

Below, we analyze the security of the session key. The lengths of session keys are 128-bit MD5 values, and the input of MD5 function is a 40-bit key-block and two 32-byte transformed random numbers that are derived from an 8-byte deal password. For less workload of exhaustive attack, the first thing adversary should do is to guess the 8-byte deal password. Then we should calculate the information quantity of the 8-byte deal password. The deal

password is passed by a short message that is shown in a mobile phone, and by referring ASCII chart, we find at most 22 special characters (such as 00H) cannot be displayed, then possible number do deal password is $2^7 - 33 = 95$, and the entropy of each character is $\log_2^{2^7-33} = 6.57$ bits. Then the random 8-byte deal password is 6.57×8=52.56 bits. Adding the 40 bits of key-block, the length of a session key in the scheme is 52.56+40=92.56 bits that are 2.314 times of the length of session key in the original SSL.

Thus, if an adversary wants to make an exhaustive attack to a session key, he has to guess the 40-bit $key-block$ and 52.56-bit deal passwords, and uses the first four characters of the "finished" message to verify the guess. The workload is much more than that of exhaustive attack to original SSL.

V. CONCLUSION

If we take no account of the usability, we could furthermore add the difficulty of exhaustive attack by adding the length of deal password. For example, when length of the deal password is 16 bytes , the entropy of transformed random number is 6.57×16=105.12 , and adding 40 bits key-block, length of the MD5's input which is secret to an adversary is 145.12 (>128) bits. And then, for less workload, the adversary has to exhaust the values of MD5 function, its output is 128 bits. It is well-known that 128- bit is secure to exhaustive attack now.

However, the security of our improved scheme has a pre-condition that is the wireless channel is secure, and an adversary could not capture the deal password by wireless method.

At present, the wireless channel of mobile phone is insecure. But, it is impossible for an adversary gets a deal password of a user whose web browser is installed a faked evil root certificate, if the adversary is not near to the place where the user is.

Research on security of wireless channel is our next work.

REFERENCES

[1] E.B.Hickman, "The SSL Protocol," Netscape Communication Corp, Feb,9th,1995.

[2] F. Lian, J. Moyne, and D. Tillbury. "Networked Design Consideration for Distributed Control Systems," IEEE Transactions on Control Systems Technology, 2002, 10(2), pp.297-307.

[3] P. Chandra, M. Messier, and J. Viega, "Network Security with OpenSSL," O'Reilly, June 2002.

[4] A. Nash, W. Duane, C. Joseph, "PKI: Implementing and Managing E-Security," Steve Burnett & Stephen palne, 2002,12.

[5] W. B. Mao, "Modern Cryptography: Theory and Practice," Publishing House of Electronics Industry, Beijing, 2004.

[6] Zhai Xuefeng, "The security analysis of SSL and the research and realization of its being hijacked," Sichuan University,2004.

[7] G. Apostolopoulos, V. Peris V,D. Saha, "Transport Layer Security How much does it really cost," Infocom'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, IEEE,1999,pp. 717-725.

[8] J. Overbey, W. Traves, and J. Wojdylo, "On the Keyspace of the Hill Cipher," Cryptologia, January 2005, 29(1), pp. 59–72.