VULNERABILITIES OF THE SSL/TLS PROTOCOL

Jelena Ćurguz

Department of IT development and services,
Post Office, Banja Luka, BiH
jelena.curguz@postesrpske.com

ABSTRACT

This paper analyzes vulnerabilities of the SSL/TLS Handshake protocol, which is responsible for authentication of the parties in the communication and negotiation of security parameters that will be used to protect confidentiality and integrity of the data. It will be analyzed the attacks against the implementation of Handshake protocol, as well as the attacks against the other elements necessary to SSL/TLS protocol to discover security flaws that were exploited, modes of attack, the potential consequences, but also studying methods of defense. All versions of the protocol are going to be the subject of the research but emphasis will be placed on the critical attack that the most endanger the safety of data. The goal of the research is to point out the danger of existence of at least vulnerability in the SSL/TLS protocol, which can be exploited and endanger the safety of the data that should be protected.

KEYWORDS

SSL/TLS Protocols, Handshake Protocols, Attacks,

1. INTRODUCTION

Internet today has become of great importance for the economy, education, business, and almost all other aspects of society, which is becoming an irreplaceable tool for work and for getting of the necessary information. Most companies organize their business via the Internet, all business communications, distribution, purchase, sale, marketing and servicing of products are made via the Internet. Environment such as e-banking, e-commerce, e-business and other offer many benefits to its users. The simplest way to perform a financial transaction is over the Internet. Use of cloud environments is becoming more popular, the possibilities it offers to its customers are very useful: access to data from anywhere, from any device, at any time.

The popularity of Internet is constantly increasing but carries with it certain security risks. A lot of data which are transmitted over the Internet infrastructure contain confidential and sensitive information (credit card number, user credentials, personal data,...) which require protection. Most corporations give their trust to SSL/TLS (Secure Sockets Layer/Transport Layer Security) protocol for data protection, which is also the most common way to protect data. However, because of its frequent use and role which it has for the protection of highly sensitive data, it is very attractive to detect and exploit security vulnerabilities.

Jan Zizka et al. (Eds) : CCSEIT, AIAP, DMDB, MoWiN, CoSIT, CRIS, SIGL, ICBB, CNSA-2016
pp. 245–256, 2016. © CS & IT-CSCP 2016 DOI : 10.5121/csit.2016.60620

2. SSL/TLS PROTOCOL

2.1. Introduction to SSL/TLS

SSL (Secure Sockets Layer), later called TLS (Transport Layer Security) is a cryptographic protocol designed to ensure the security of data transmitted over the Internet. Developed by the Netscape company in 1994 and in 1996 the company issued the latest version of this protocol called SSL3.0. Further development and release of the protocol took over the IETF organization but later named TLS. The protocol is used to protect the data on the transport layer.

It is located between the transport and application layer in the ISO/OSI reference model and provides security services for any application-based protocols, such as HTTP, FTP, LDAP, POP3,... It is used in client/server environment and provides following features for parties in communication:

- authentication,
- confidentiality
- integrity

2.2. Structure of the SSL/TLS protocol

SSL/TLS protocol consists of two layers and several protocols. The lower layer located next to the transport lavel in the OSI/ISO reference model consists of SSL/TLS Record protocol. The higher layer located immediately above the Record protocol consists of the SSL/TLS Handshaking protocols: Handshake protocol, ChangeCipherSpec protocol and Alert protocol. (Figure 1.)

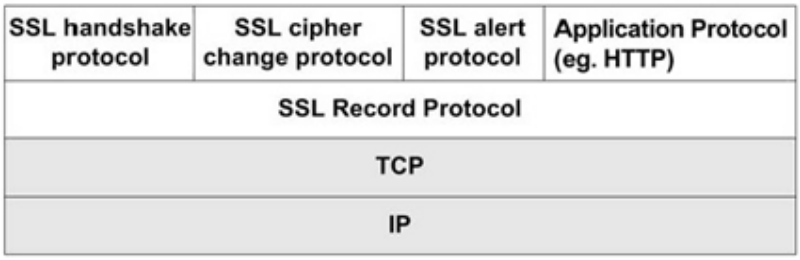

Figure 1. SSL/TLS protocol

2.2.1 Record protocol

The Record protocol is responsible for the transfer of blocks of data between the two sides in communication. It takes messages from application level of the OSI/ISO reference models, divides them into manageable blocks, optionally compresses, applies MAC, encrypts and transmits results. It uses security parameters negotiated during handshake phase.

2.2.2 Handshake protocol

The Handshake protocol is the core protocol of SSL/TLS responsible for authentication of each party of the communication and negotiation of security parameters to be used for exchange of encrypted data.

2.2.3. ChangeCipherSpec protocol

The ChangeCipherSpec protocol is used to notify both parties in the communication to upgrade the status of the session to negotiated parameters and move on to secure communication.

2.2.4. Alert protocol

The Alert protocol is used for the notification of errors that occur in communication between the two sides, i.e.: when the connection is closed, when the message can not be decrypted, etc..
During the handshake phase all cryptographic primitives responsible for connection protection are established. Communication between client and server during handshake phase is done with the messages with predefined forms. One example of exchanged messages between client and server during the handshake phase can be seen in Figure 2. The messages are:

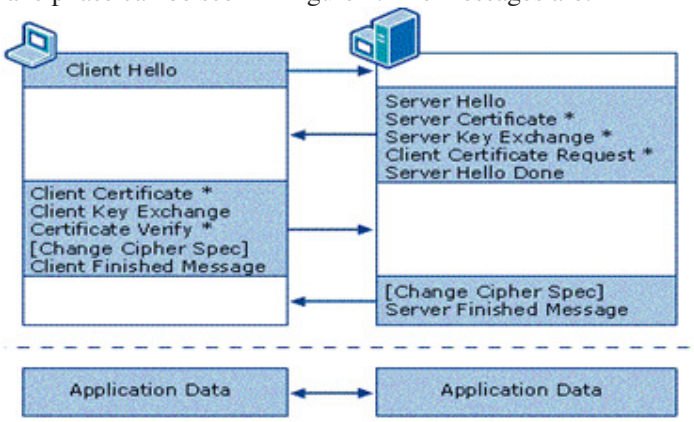

Figure 2. Messages of the Handshake protocol

1. ClientHello is type of message where the client notifies the server of the security parameters (protocol version, client random, session ID, cipher suite, compression method) which he supports and wants to use.

2. ServerHello is a message where the server notifies the client of the security parameters that will be used (protocol version, server random, cipher suite, session ID, compression method);

3. Certificate is a message which transmitted certificate for server and optional message for transmitted client's certificate.

4. ServerKeyExchange message is following ServerHello message if using anonymous negotiation or server Certificate message if there are not enough informations for the client in order to exchange premaster secret to the server. Optional message.

5. CertificateRequest message is a message where the server requires certificate from the client. Optional message.

6. ServerHelloDone message is a message where the server notifies that all requests are sent to the client in order to establish a communication.

7. ClientKeyExchange message contains generated premaster key encrypted with server's public key. Based on the premaster secret, later will be generated a master secret and based on the master secret will be generated all other keys in order to encrypt the traffic.

8. CertificateVerify message is a message where the client confirms that he has a private key corresponding to the public key from the certificate. This message is sent only if the client previously sent a client Certificate message. Optional message.

9. ChangeCipherSpec message is a type of message where the other party in communication is informed about the start of the use of agreed security settings. After these messages all the other messages that are exchanged are encrypted.
10. Finished message informs that all the steps of negotiations are done and that a secure communication is established.

When the process of negotiation of parameters and generating of necessary keys is finished, Record protocol takes the data from the application level, decomposes it into smaller blocks, optionally compresses it, applies the agreed hash function, encrypts with the agreed algorithm and then transport it through the transport layer to the other side in communication.

3. ATTACKS ON THE SSL/TLS PROTOCOL

SSL/TLS protocol is a widespread used protocol for data protection. However, because of that, it is very interesting for discovering and exploiting security flaws that harm the integrity and security of data. This paper will briefly analyze some of the attacks on SSL/TLS Handshake protocol, as well as other elements required in the work protocols, to analyze security flaws that were exploited, modes of attack, the potential consequences, but also studying methods of defense. The attacks, which will be discussed are:

- Ciphersuite rollback attack
- Drop ChangeCipherSpec attack
- Version rollback attack
- Key Exchage Algorithm confusion
- Bleichenbacher attack on PKCS#1
- Attacking RSA-Based Sessions in SSL/TLS
- Timing attack
- Cross-protocol attack based on ECC key exchange
- New Bleichenbacher side channels and attacks
- SKIP-TLS attack
- FREAK atack
- Logjam attack

3.1. CipherSuite rollback attack

Cipher suite is a list of cryptographic algorithms that are proposed during the handshake phase between the client and server. List of proposed algorithms is traveling in clean text format as part of the initial ClientHello messages. It allows MITM attacker to intercept the message and replacing client's cipher suite with his cipher suite that supports weaker versions of algorithms or NULL-Cipher list, so communication continues to take place with weaker algorithms or algorithms for protection are not used at all [2]. The consequences of such an attack could be disastrous for the client: the attacker could imitate a valid user, could access the server, obtain user credentials and the like.

In version SSL3.0 this failure is resolved with the authentication of all handshake messages in the final Finished message, which contains MAC on handshake protocol messages, keyed by the

master secret, so the attack on cipher suite might be noticed at the end of the handshake phase and reject such a session [2].

3.2. Drop ChangeCipherSpec attack

Drop ChangeCipherSpec attack is weakness of SSL2.0 discovered by David Wagner and Bruce Schneier. The ChangeCipherSpec message is used to notify both parties in the communication to upgrade the status of the session to negotiated parameters in the handshake phase. When the initial handshake phase is completed, the client and server exchange ChangeCipherSpec message to signal the other side that in the future all communication will be done only with the agreed parameters. However, before one side send the ChangeCipherSpec message MITM attacker can send Finished message to other side, which furthermore would cause the start of communication without any changes and adoptions of agreed security parameters, or it could simply delete ChangeCipherSpec message, so the client and the server would never establish a communication [2].

The solution for this problem is to force both parties to ensure that a ChangeCipherSpec message is received before accepting the Finished message. RFC 2246 says "It is essential that a ChangeCipherSpec message is received between the other handshake messages and prior to the Finished message" [5].

3.3. Version rollback attack

Version rollback attack is a vulnerability of SSL 3.0. It is a type of attack where the attacker can lead the client and server side of the communication to use a lower version of the protocol SSL2.0 instead of one they should use and support SSL3.0. The attacker modifies ClientHello message SSL3.0, so it looks like ClientHello message SSL2.0. In that way it can use the weaknesses of protocol SSL2.0, like weaker cipher suite of proposed algorithms (i.e. a list with the DES algorithm, which further allows an attacker brute force attack and compromising of sensitive data).

Paul Kocher has designed a strategy to detect version rollback attacks so that clients that support SSL3.0 have built-in fixed redundancy in the RSA PKCS padding bytes in order to indicate that they support SSL3.0. Servers will refuse SSL2.0 connection if they notice that these bytes are present on the client side [2]. Unfortunately, this strategy is valid only for RSA key exchange but not for Diffie Hellman. SSL 3.0 and later TLS versions offer protection for version rollback attacks with the Finished message (ClientHello message contains the client_version field, which shows its highest supported version). Also in case RSA, the client generates a 48-byte premaster secret, and the version number in the premaster secret is the version offered by the client in the Client Hello, not the version negotiated for the connection.

3.4. Key Exchange Algorithm confusion or Cross-protocol attack

Key Exchange Algorithm confusion or Cross-protocol attack is vulnerability of SSL3.0. Server can send to the client the temporary key parameters signed under its long-term certified signing key in ServerKeyExchange messages [2]. The problem is that the signature of the temporary key parameters does not include part of the field where it is specified which type of key is used, the RSA or Diffie Hellman, and thus created a basis for a confusion type of attack.

The attacker forced the server to use the Diffie-Hellman key exchange and client to use RSA key exchange. This leads to confusion where the client may interpret the Diffie-Hellman parameters

(p, g) as an exponent and module of RSA key. In the following example we can see how the attack works [2].

[ClientHello]
Client ->Attacker: SSL_RSA…
Attacker->Server: SSL_DHE_RSA…
[ServerHello]
Server->Attacker: SSL_DHE_RSA…
Attacker->Client: SSL_RSA…
[ServerKeyExchange]
Server->Attacker: $\{p,g,y\}K_s$
Attacker->Client: $\{p,g,y\}K_s$
[ClientKeyExchange]
Client->Attacker: $k^g \bmod p$
Attacker->Server: $g^x \bmod p$

$k^g \bmod p$, k is premaster secret. For successful attack the client intercept premaster key which will be encrypted with RSA key, or Diffie Hellman parameters *(g, p)* which attacker already knows. Furthermore, the attacker sends to server $g^x \bmod p$ whereby the server interprets premaster key as $g^{xy} \bmod p$. Attacker in future can intercept, read and change all the exchanged messages between the client and the server, act as a server to the client and as a client to the server.

Proposed solution for cross protocol attack was a new protocol extension indicating the new format of ServerKeyExchange message which includes explicit indicators of the entity (server), the type of key exchange algorithm, the handshake messages exchanged and the parameters of the key exchange [4].

3.5. Bleichenbacher attack on PKCS#1

Daniel Bleichenbacher in 1999 presented the attack where it is possible for a certain amount of time to decript premaster secret encrypted with server's RSA public key. Premaster secret encrypted with RSA algorithm is the value generated by the client and sent to the server (encrypted and formatted with the standard PKCS v1.5) within ClientKeyExchange messages.
To encrypt a message M, with the RSA public key in accordance with the standard PKCS v1.5 format of message must be

0x00 0x02 [non-zero bytes] 0x00 [M]

At the beginning of the message always comes *0x00 0x02* bytes, after padding bytes, *0x00,* and finally message *M*.

The attack is based on the chosen ciphertext attack. The core of Bleichenbacher's attack relies on an *oracle*: the attack works if there is some system, that for any chosen ciphertext *C*, indicates whether the corresponding plaintext has the correct format according to the standard PKCS#1 [8]. Scenario for an attack is the following:

- The attacker has access to a system (*oracle*) that for each selected encrypted data returns the value *true* or *false* depending on whether the decrypted message is or not in accordance with PKCS#1 v1.5 structure.

- The attacker eavesdropps the communication and wants to decrypt the encrypted information *C*. He knows that $C = M^e \bmod n$. He wants to calculate $M = C^d \bmod n$, or the premaster secret.

Attacker sends a large number of requests with a random value and requires from the system (*oracle*) to decrypt messages $C' = s^e C \bmod n$. System decrypt the message as $M' = (C')^d \bmod n$.(Figure 3)

Based on multiple responses M' which are in accordance with PKCS#1 v1.5 standard, attacker concludes about the possible values of M and decrypts the message as $M = M's^{-1} \bmod n$ [8].

If the oracle answers with "true", the attacker knows that messages M start with *0x00 0x02* and $2B < M < 3B - 1$, where $B = 2^{8(k-2)}$ and k bytes length of n. If the oracle responds with "false", the attacker increments s and repeats request to oracle. By iteratively choosing new values for s, querying the oracle, the attacker narrows down the interval which contains the original M value.

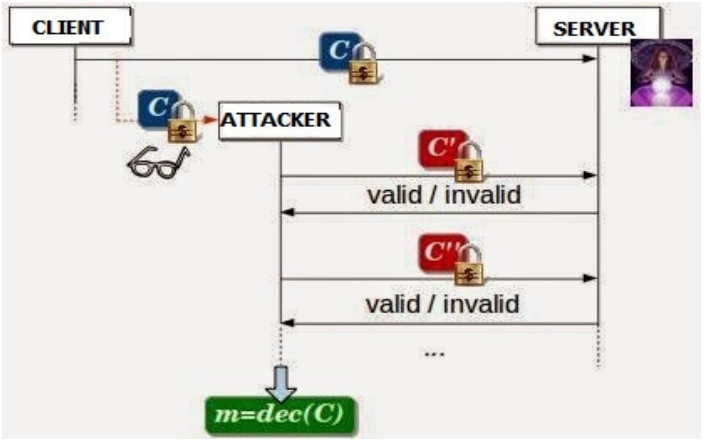

Figure 3. Bleichenbacher attack

Possible defense from the attack is that the server does not inform the client of irregular PSCS#1.5 format of message. The RFC 2246 says: "The best way to avoid vulnerability to this attack is to treat incorrectly formatted messages in a manner indistinguishable from correctly formatted RSA blocks. Thus, when it receives an incorrectly formatted RSA block, a server should generate a random 48-byte value and proceed using it as the premaster secret. Thus, the server will act identically whether the received RSA block is correctly encoded or not" [5].

3.6. Attacking RSA-Based Sessions in SSL/TLS

In 2003 the researchers Klima, Pokorny and Rosa presented "bad version oracle" attack. A countermeasure from the Bleichenbacher's attack from 1999 is to generate a random premaster secret and continue with the handshake phase until the verification and decryption of the Finished message fails, due to different key material.

Encrypted data of the ClientKeyExchange message in an RSA-based handshake includes also the major and minor version number of protocol offered by the client. Format of message is

0x00 0x02 [non-zero bytes] 0x00 [0x03 0x01 Random baytes]

where *[0x03 0x01 Random baytes]* is a message M or premaster secret and *0x03 0x01* means TLS1.0 or SSL3.1.

Many implementations checked for equality of the received protocol version contained in the ClientKeyExchange message to the one expected. This check requires a valid PKCS#1 v1.5 structure. Unfortunately, it has not been specified how such a check may be combined with the

countermeasure from the Bleichenbacher's attack and therefore creates the basis for build a "bad version oracle" attack. Form of attacks is an extended version of Bleichenbacher's attack.

$$O_{BadVersion}(x) = \begin{cases} true, & if\ version\ number\ is\ valid \\ false, & otherwise \end{cases}$$

In case of protocol version mismatch an Alert message was returned to the sender and sender knows that message is in accordance with PKCS#1 structure and starts with *0x00 0x02*.

The authors propose to keep generating premaster secret randomly if messages is not PKCS conforming. They propose to replace major and minor version number of protocol with the expected version number in either case (i.e. if message is or is not PKCS-conforming) [9]. Furthermore, RFC 5246 says "In any case, a TLS server MUST NOT generate an alert if processing an RSA-encrypted premaster secret message fails, or the version number is not as expected. Instead, it MUST continue the handshake with a randomly generated premaster secret"[10].

3.7. Timing attack

Timing attack is a form of side channel attack that exploits time to reveal the encrypted data. Attack exploits timing variants of cryptographic operations for different values of the input data. This attack computes the private key on the server by calculating the time difference between sending a specially made ClientKeyExcange messages and receiving Alert message that alerts the irregular premaster secret [3].

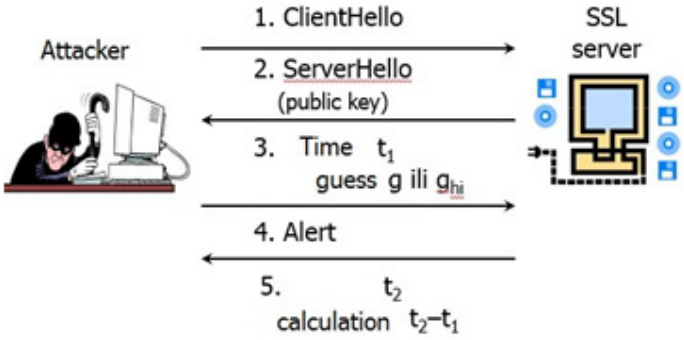

Figure 4. Timing attack

Encryption of message M with RSA algorithm is $C = M^e \bmod n$, and decryption is $M = C^d \bmod n$, where (e,n) is RSA public key and d is private RSA key. The goal is to find the private key d [6]. The attacker must have access to the target system that will count $C^d \bmod n$ for a few selected values of C. (Figure 4) With the precise count of time and with analyzing different time variants, with sending ClientKeyExcahge messages (different values of C) and receiving Alert messages, the attacker calculates the individual bits of the private key $d = d0d1\ldots$.

The most widely used method for defense from this attack is RSA blinding. Because the timing attacks expose information by measuring the amount of time required to perform mathematical operations, before decrypting the ciphertext C system first compute $X = r^e C \bmod n$, where r is a random value. For decrypt X system compute $X^d \bmod n = r^{ed}\ C^d \bmod n = r\ C^d \bmod n$. Then the output multiply by r^{-1} to obtain $M=C^d \bmod n$ which is the plaintext. Since a different r is used for each message, blinding prevents an attacker from computation time variant operations during decoding [11].

3.8. Cross-protocol attack based on ECC (Elliptic Curve Cryptography) key exchange

In 2012 serveral researchers discovered new cross protocol attack similar cross protocol attack from 2003. The main idea is the same as in the earlier attack, but now at the client side the attacker will negotiate ephemeral Diffie-Hellman key exchange, and on the server side ephemeral Elliptic Curve Diffie-Hellman key exchange. The attacker will intercept the TLS handshake messages between the server and the client and alter some of them that the client thinks that it is used ephemeral Diffie-Hellman key exchange and server ephemeral Elliptic Curve Diffie-Hellman key exchange. The goal of an attacker is to impersonates a server to the client and puts itself in between them, for the opportunity to reads, changes and forwards all the messages [13].
Proposed solution for cross protocol attack was a new protocol extension indicating the new format of ServerKeyExchange message which includes explicit indicators of the entity (server), the type of key exchange algorithm, the handshake messages exchanged and the parameters of the key exchange [4].

3.9. New Bleichenbacher side channels and attacks

In 2014 researchers presented a few new Bleichenbacher side channels and attack. They present four new Bleichenbacher's side channels and three successful Bleichenbacher attacks against the JSSE (SSL/TLS implementation) and against hardware security appliances using the Cavium NITROX SSL accelerator chip. These attacks are: error messages in JSSE, timing differences in JSSE, GnuTLS and OpenSSL implementations, internal exception in JSSE and unexpected timing behavior by hardware appliances [14].

For all these vulnerabilities researchers were found patches.

3.10. SKIP-TLS: Message Skipping Attacks on TLS

SKIP-TLS is a set of vulnerabilties found in popular open source TLS implementations (OpenSSL, GnuTLS, CyaSSL, JSSE,...) during 2015. Researchers tested popular open source implementations for state machine bugs and discover several new critical security vulnerabilities. They find that several TLS implementations incorrectly allow some handshake messages to be skipped even though they are required for current handshake [15].

Patches for these vulnerabilities are in development.

3.11. Freak (Factoring Attack on RSA-EXPORT Keys) attack

The attack discovered in March 2015 exploits the weaknesses of some implementations of SSL/TLS protocols that supports the "export-grade cryptography" [16]. Attacker by MITM attack proposes to server the use of RSA_EXPORT cipher suite that uses weaker RSA keys, i.e. keys of 512 bits or even less. Server accepting such a demand endangers the detection of weak RSA key and decrypion of the traffic. The vulnerabiliy was due to the previous law of the US government where it didn't allow to export strong cryptographic algorithms in other countries.

Defense against this attack is to forbid cryptography based on weak cryptographic algorithms created under the former law of the US government.

3.12. Logjam attack

The attack discovered in June 2015 is similar to Freak attack, because it is based on the already

well-known weaknesses of the TLS protocol that supports cryptography arising from the previous law of US government for the other countries, but this time was not directed at the RSA key exchange but on the Diffie-Hellman.

Diffie-Hellman key exchange is a widespread method of securely exchanging cryptographic keys. It is main method for the SSH and IPsec protocol and one of the options for TLS. The way to generate secret keys works as follows: Alice and Bob agree on a primary number p and a generator g. Alice sends $g^a \ mod \ p$ to Bob and Bob sends $g^b \ mod \ p$ to Alice, but each of them calculates the secret key $g^{ab} \ mod \ p$. The best technique for the attack on the Diffie-Hellman key exchange is based on the compromising the one of the private exponent *(a, b)* calculating the discreet log. The attacker who can find the discrete log x of $y = g^x \ mod \ p$ easily calculate the private key [17].

Previous law of the US government would not allow export of the algorithm in which the prime number was greater than 512 bits. Logjam attack is MITM attack in which an attacker can do downgrade TLS connection to the 512-bit "export-grade cryptography" connection. As we know, the client sends a list of proposed algorithms to server within ClientHello messages and based on that server chooses a list and indicate its choice in ServerHello message. Protocol supports several different variants of Diffie-Hellman key exchange: ephemeral, fixed and anonymous Diffie-Hellman key exchange.[1] The ephemeral Diffie-Hellman key exchange is mostly used.[1] At ephemeral Diffie-Hellman key exchange server is responsible for the selection of the parameters *(p, g),* and it calculates $g^b \ mod \ p$ and sends ServerKeyExchange message that contains the signature of the selected parameters (p, g, g^b) by signing key. The client verifies the signature and responds with ClientKeyExchange message that contains g^a. Both sides, on the basis of shared parameters, are calculating a master secret as $g^{ab} \ mod \ p$ and calculates a MAC of all exchanged handshake messages and exchanges them in the Finished message.[17] After calculating the encryption keys client and server may begin secure communication. The attacker, who is placed in the middle of communication, intercepts ClientHello message and removes all other lists of proposed algorithms and instead all DHE lists of algorithms puts DHE_EXPORT list that server accepts, if it could support it. Then it waits for ServerHello message where it also changes the DHE_EXPORT into DHE list and forwards ServerKeyExchange message. The client will interpret the DHE_EXPORT parameters (p, g, g^b) as valid DHE parameters selected by the server and will continue with the handshake phase. The attacker who can calculate secret *b,* from ClientKeyExchange messages, in the given time and based on that it can calculate the master secret and the encryption keys, so it can finish the handshake phase with the client in due time. After that he can continue reading and changing data with the client imitating the server [17].
Defense against this attack is to forbid cryptography based on weak cryptographic algorithms created under the former law of the US government.

4. CONCLUSION

Although, in this paper we didn't analyse all the attacks on SSL/TLS Handshake protocol, it can be concluded that in the long history of the SSL/TLS protocol, exactly this part of the SSL/TLS protocol has been exposed to various types of attacks. Negotiation of security parameters is the most critical part in work of the SSL/TLS protocol, whose compromising completely endangers the security of data transmitted to transport layer. Previous attacks on SSL/TLS protocol took the adventage of the fact that there was no authentication of the messages during handshake phase, no verification in the order of arrival of the handshake messages, used weakness of PKCS#1 v1.5, used some form of side channel attacks (time) and the like. Most of these attacks are implementation weaknesses of SSL/TLS protocol, while others are weaknesses in other elements used in work protocol. However, 20 years later researchers have been still finding some weaknesses in the protocol such as state machine bugs, support "export-grade cryptography",

Bleichenbacher side channels attacks and other.

The conclusion based on the our research is that the maximum attention was devoted to the entire mode and security of Handshake protocol and maximum effort was made to find ways of a defense from previous attacks. However, as any other sistem, so the SSL/TLS protocol can not be completely safe and there is always the danger of finding new vulnerabilities and their exploitation. Because of its large role in the protection of highly sensitive data from all attacks on SSL/TLS protocol, whether they are directed to the protocol itself or to some other elements, it is necessary to find a solution for the defense and to eliminate security flaw.

REFERENCES

[1] Rolf Oppliger, "SSL and TLS: Theory and Practice", 2009

[2] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In Proceedings of the 2nd USENIX Workshop on Electronic Commerce, volume 2 of WOEC, pages 29–40. USENIX Association, 1996

[3] Christopher Meyer and Jorg Schwenk , "Lessons Learned From SSL/TLS Attacks",

[4] N. Mavrogiannopoulos "Preventing cross-protocol attacks in TLS protocol draft-mavrogiannopoulos-tls-server-key-exchage-00", 2012

[5] RFC 2246, "The TLS Protocol Version 1.0", 1999

[6] http://crypto.stackexchange.com/questions/12688/can-you-explain-bleichenbachers-cca-attack-on-pkcs1-v1-5

[7] Christopher Meyer, "20 Years of SSL/TLS Research An Analysis of the Internet's Security Foundation", 2014

[8] Bleichenbacher, D., "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1," Proceedings of CRYPTO '98, Springer-Verlag, LNCS 1462, August 1998,

[9] Vlastimil Klíma, Ondrej Pokorný and Tomáš Rosa, "Attacking RSA-Based Sessions in SSL/TLS", 2003

[10] RFC 5246 The Transport Layer Security (TLS) Protocol Version 1.2

[11] http://www.cs.sjsu.edu/faculty/stamp/students/article.html#authorbio,"Timing Attacks on RSA", Wing.H.Wong

[12] https://en.wikipedia.org/wiki/Side-channel_attack

[13] Andrija Jakovljević, "Exploring cross-protocol attacks on the TLS protocol"

[14] Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, Erik Tews "Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks", 2014

[15] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournety, Markulf Kohlweissy, Alfredo Pironti, Pierre-Yves Strubz, Jean Karim Zinzindohoue "Taming the Composite State Machines of TLS", 2015

[16] https://www.smacktls.com/, "Factoring RSA Export Keys"

[17] David Adrian, Karthikeyan Bhargavan,.. "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice".

AUTHORS

Jelena Ćurguz has been employed in the Post of Republic of Srpska in Banja Luka, Bosnia and Herzegovina since 2003. She has been a Head of the Service for system support since 2008. She acquired extensive experience in the maintenance of systems and services in the Linux platform, Internet services, Informix database etc. She owns certificates in the field of Red Hat Linux. She is attending postgraduate studies at the Faculty of Electrical Engineering in Banja Luka.