Appendix F. Security Analysis

The TLS protocol is designed to establish a secure connection between a client and a server communicating over an insecure channel. This document makes several traditional assumptions, including that attackers have substantial computational resources and cannot obtain secret information from sources outside the protocol. Attackers are assumed to have the ability to capture, modify, delete, replay, and otherwise tamper with messages sent over the communication channel. This appendix outlines how TLS has been designed to resist a variety of attacks.

F.1. Handshake Protocol

The handshake protocol is responsible for selecting a cipher spec and generating a master secret, which together comprise the primary cryptographic parameters associated with a secure session. The handshake protocol can also optionally authenticate parties who have certificates signed by a trusted certificate authority.

F.1.1. Authentication and Key Exchange

TLS supports three authentication modes: authentication of both parties, server authentication with an unauthenticated client, and total anonymity. Whenever the server is authenticated, the channel is secure against man-in-the-middle attacks, but completely anonymous sessions are inherently vulnerable to such attacks. Anonymous servers cannot authenticate clients. If the server is authenticated, its certificate message must provide a valid certificate chain leading to an acceptable certificate authority. Similarly, authenticated clients must supply an acceptable certificate to the server. Each party is responsible for verifying that the other's certificate is valid and has not expired or been revoked.

The general goal of the key exchange process is to create a pre_master_secret known to the communicating parties and not to attackers. The pre_master_secret will be used to generate the master_secret (see Section 8.1). The master_secret is required to generate the Finished messages, encryption keys, and MAC keys (see Sections 7.4.9 and 6.3). By sending a correct Finished message, parties thus prove that they know the correct pre_master_secret.

F.1.1.1. Anonymous Key Exchange

Completely anonymous sessions can be established using Diffie-Hellman for key exchange. The server's public parameters are contained in the server key exchange message, and the client's are sent in the

client key exchange message. Eavesdroppers who do not know the private values should not be able to find the Diffie-Hellman result (i.e., the pre_master_secret).

Warning: Completely anonymous connections only provide protection against passive eavesdropping. Unless an independent tamper-proof channel is used to verify that the Finished messages were not replaced by an attacker, server authentication is required in environments where active man-in-the-middle attacks are a concern.

F.1.1.2. RSA Key Exchange and Authentication

With RSA, key exchange and server authentication are combined. The public key is contained in the server's certificate. Note that compromise of the server's static RSA key results in a loss of confidentiality for all sessions protected under that static key. TLS users desiring Perfect Forward Secrecy should use DHE cipher suites. The damage done by exposure of a private key can be limited by changing one's private key (and certificate) frequently.

After verifying the server's certificate, the client encrypts a pre_master_secret with the server's public key. By successfully decoding the pre_master_secret and producing a correct Finished message, the server demonstrates that it knows the private key corresponding to the server certificate.

When RSA is used for key exchange, clients are authenticated using the certificate verify message (see Section 7.4.8). The client signs a value derived from all preceding handshake messages. These handshake messages include the server certificate, which binds the signature to the server, and ServerHello.random, which binds the signature to the current handshake process.

F.1.1.3. Diffie-Hellman Key Exchange with Authentication

When Diffie-Hellman key exchange is used, the server can either supply a certificate containing fixed Diffie-Hellman parameters or use the server key exchange message to send a set of temporary Diffie-Hellman parameters signed with a DSA or RSA certificate. Temporary parameters are hashed with the hello.random values before signing to ensure that attackers do not replay old parameters. In either case, the client can verify the certificate or signature to ensure that the parameters belong to the server.

If the client has a certificate containing fixed Diffie-Hellman parameters, its certificate contains the information required to complete the key exchange. Note that in this case the client and server will generate the same Diffie-Hellman result (i.e.,

pre_master_secret) every time they communicate. To prevent the pre_master_secret from staying in memory any longer than necessary, it should be converted into the master_secret as soon as possible. Client Diffie-Hellman parameters must be compatible with those supplied by the server for the key exchange to work.

If the client has a standard DSA or RSA certificate or is unauthenticated, it sends a set of temporary parameters to the server in the client key exchange message, then optionally uses a certificate verify message to authenticate itself.

If the same DH keypair is to be used for multiple handshakes, either because the client or server has a certificate containing a fixed DH keypair or because the server is reusing DH keys, care must be taken to prevent small subgroup attacks. Implementations SHOULD follow the guidelines found in [SUBGROUP].

Small subgroup attacks are most easily avoided by using one of the DHE cipher suites and generating a fresh DH private key (X) for each handshake. If a suitable base (such as 2) is chosen, g^X mod p can be computed very quickly; therefore, the performance cost is minimized. Additionally, using a fresh key for each handshake provides Perfect Forward Secrecy. Implementations SHOULD generate a new X for each handshake when using DHE cipher suites.

Because TLS allows the server to provide arbitrary DH groups, the client should verify that the DH group is of suitable size as defined by local policy. The client SHOULD also verify that the DH public exponent appears to be of adequate size. [KEYSIZ] provides a useful guide to the strength of various group sizes. The server MAY choose to assist the client by providing a known group, such as those defined in [IKEALG] or [MODP]. These can be verified by simple comparison.

F.1.2. Version Rollback Attacks

Because TLS includes substantial improvements over SSL Version 2.0, attackers may try to make TLS-capable clients and servers fall back to Version 2.0. This attack can occur if (and only if) two TLS- capable parties use an SSL 2.0 handshake.

Although the solution using non-random PKCS #1 block type 2 message padding is inelegant, it provides a reasonably secure way for Version 3.0 servers to detect the attack. This solution is not secure against attackers who can brute-force the key and substitute a new ENCRYPTED-KEY-DATA message containing the same key (but with normal padding) before the application-specified wait threshold has expired. Altering the padding of the least-significant 8 bytes of the PKCS

padding does not impact security for the size of the signed hashes and RSA key lengths used in the protocol, since this is essentially equivalent to increasing the input block size by 8 bytes.

F.1.3. Detecting Attacks Against the Handshake Protocol

An attacker might try to influence the handshake exchange to make the parties select different encryption algorithms than they would normally choose.

For this attack, an attacker must actively change one or more handshake messages. If this occurs, the client and server will compute different values for the handshake message hashes. As a result, the parties will not accept each others' Finished messages. Without the master_secret, the attacker cannot repair the Finished messages, so the attack will be discovered.

F.1.4. Resuming Sessions

When a connection is established by resuming a session, new ClientHello.random and ServerHello.random values are hashed with the session's master_secret. Provided that the master_secret has not been compromised and that the secure hash operations used to produce the encryption keys and MAC keys are secure, the connection should be secure and effectively independent from previous connections. Attackers cannot use known encryption keys or MAC secrets to compromise the master_secret without breaking the secure hash operations.

Sessions cannot be resumed unless both the client and server agree. If either party suspects that the session may have been compromised, or that certificates may have expired or been revoked, it should force a full handshake. An upper limit of 24 hours is suggested for session ID lifetimes, since an attacker who obtains a master_secret may be able to impersonate the compromised party until the corresponding session ID is retired. Applications that may be run in relatively insecure environments should not write session IDs to stable storage.

F.2. Protecting Application Data

The master_secret is hashed with the ClientHello.random and ServerHello.random to produce unique data encryption keys and MAC secrets for each connection.

Outgoing data is protected with a MAC before transmission. To prevent message replay or modification attacks, the MAC is computed from the MAC key, the sequence number, the message length, the

message contents, and two fixed character strings. The message type field is necessary to ensure that messages intended for one TLS record layer client are not redirected to another. The sequence number ensures that attempts to delete or reorder messages will be detected. Since sequence numbers are 64 bits long, they should never overflow. Messages from one party cannot be inserted into the other's output, since they use independent MAC keys. Similarly, the server write and client write keys are independent, so stream cipher keys are used only once.

If an attacker does break an encryption key, all messages encrypted with it can be read. Similarly, compromise of a MAC key can make message-modification attacks possible. Because MACs are also encrypted, message-alteration attacks generally require breaking the encryption algorithm as well as the MAC.

Note: MAC keys may be larger than encryption keys, so messages can remain tamper resistant even if encryption keys are broken.

F.3. Explicit IVs

[CBCATT] describes a chosen plaintext attack on TLS that depends on knowing the IV for a record. Previous versions of TLS [TLS1.0] used the CBC residue of the previous record as the IV and therefore enabled this attack. This version uses an explicit IV in order to protect against this attack.

F.4. Security of Composite Cipher Modes

TLS secures transmitted application data via the use of symmetric encryption and authentication functions defined in the negotiated cipher suite. The objective is to protect both the integrity and confidentiality of the transmitted data from malicious actions by active attackers in the network. It turns out that the order in which encryption and authentication functions are applied to the data plays an important role for achieving this goal [ENCAUTH].

The most robust method, called encrypt-then-authenticate, first applies encryption to the data and then applies a MAC to the ciphertext. This method ensures that the integrity and confidentiality goals are obtained with ANY pair of encryption and MAC functions, provided that the former is secure against chosen plaintext attacks and that the MAC is secure against chosen-message attacks. TLS uses another method, called authenticate-then-encrypt, in which first a MAC is computed on the plaintext and then the concatenation of plaintext and MAC is encrypted. This method has been proven secure for CERTAIN combinations of encryption functions and MAC functions, but it is not guaranteed to be secure in general.

In particular, it has been shown that there exist perfectly secure encryption functions (secure even in the information-theoretic sense) that combined with any secure MAC function, fail to provide the confidentiality goal against an active attack. Therefore, new cipher suites and operation modes adopted into TLS need to be analyzed under the authenticate-then-encrypt method to verify that they achieve the stated integrity and confidentiality goals.

Currently, the security of the authenticate-then-encrypt method has been proven for some important cases. One is the case of stream ciphers in which a computationally unpredictable pad of the length of the message, plus the length of the MAC tag, is produced using a pseudorandom generator and this pad is exclusive-ORed with the concatenation of plaintext and MAC tag. The other is the case of CBC mode using a secure block cipher. In this case, security can be shown if one applies one CBC encryption pass to the concatenation of plaintext and MAC and uses a new, independent, and unpredictable IV for each new pair of plaintext and MAC. In versions of TLS prior to 1.1, CBC mode was used properly EXCEPT that it used a predictable IV in the form of the last block of the previous ciphertext. This made TLS open to chosen plaintext attacks. This version of the protocol is immune to those attacks. For exact details in the encryption modes proven secure, see [ENCAUTH].

F.5. Denial of Service

TLS is susceptible to a number of denial-of-service (DoS) attacks. In particular, an attacker who initiates a large number of TCP connections can cause a server to consume large amounts of CPU for doing RSA decryption. However, because TLS is generally used over TCP, it is difficult for the attacker to hide his point of origin if proper TCP SYN randomization is used [SEQNUM] by the TCP stack.

Because TLS runs over TCP, it is also susceptible to a number of DoS attacks on individual connections. In particular, attackers can forge RSTs, thereby terminating connections, or forge partial TLS records, thereby causing the connection to stall. These attacks cannot in general be defended against by a TCP-using protocol. Implementors or users who are concerned with this class of attack should use IPsec AH [AH] or ESP [ESP].

F.6. Final Notes

For TLS to be able to provide a secure connection, both the client and server systems, keys, and applications must be secure. In addition, the implementation must be free of security errors.

The system is only as strong as the weakest key exchange and authentication algorithm supported, and only trustworthy cryptographic functions should be used. Short public keys and anonymous servers should be used with great caution. Implementations and users must be careful when deciding which certificates and certificate authorities are acceptable; a dishonest certificate authority can do tremendous damage.

Normative References

[AES] National Institute of Standards and Technology, “Specification for the Advanced Encryption Standard (AES)” FIPS 197. November 26, 2001.

[3DES] National Institute of Standards and Technology, “Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher”, NIST Special Publication 800-67, May 2004.

[DSS] NIST FIPS PUB 186-2, “Digital Signature Standard”, National Institute of Standards and Technology, U.S. Department of Commerce, 2000.

[HMAC] Krawczyk, H., Bellare, M., and R. Canetti, “HMAC: Keyed- Hashing for Message Authentication”, RFC 2104, February 1997.

[MD5] Rivest, R., “The MD5 Message-Digest Algorithm”, RFC 1321, April 1992.

[PKCS1] Jonsson, J. and B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1”, RFC 3447, February 2003.

[PKIX] Housley, R., Polk, W., Ford, W., and D. Solo, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, RFC 3280, April 2002.

[SCH] B. Schneier. “Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed.”, Published by John Wiley & Sons, Inc. 1996.

[SHS] NIST FIPS PUB 180-2, “Secure Hash Standard”, National Institute of Standards and Technology, U.S. Department of Commerce, August 2002.

[REQ] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997.

[RFC2434] Narten, T. and H. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs”, BCP 26, RFC 2434, October 1998.

[X680] ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.

[X690] ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002, Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

Informative References

[AEAD] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[BLEI] Bleichenbacher D., "Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1" in Advances in Cryptology – CRYPTO'98, LNCS vol. 1462, pages: 1-12, 1998.

[CBCATT] Moeller, B., "Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures", http://www.openssl.org/~bodo/tls-cbc.txt.

[CBCTIME] Canvel, B., Hiltgen, A., Vaudenay, S., and M. Vuagnoux, "Password Interception in a SSL/TLS Channel", Advances in Cryptology – CRYPTO 2003, LNCS vol. 2729, 2003.

[CCM] "NIST Special Publication 800-38C: The CCM Mode for Authentication and Confidentiality", http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C.pdf

[DES] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46-3, October 1999.

[DSS-3] NIST FIPS PUB 186-3 Draft, "Digital Signature Standard", National Institute of Standards and Technology, U.S. Department of Commerce, 2006.

[ECDSA] American National Standards Institute, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANS X9.62-2005, November 2005.

[ENCAUTH] Krawczyk, H., "The Order of Encryption and Authentication for Protecting Communications (Or: How Secure is SSL?)", Crypto 2001.

[ESP] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[FI06] Hal Finney, "Bleichenbacher's RSA signature forgery based on implementation error", ietf-openpgp@imc.org mailing list, 27 August 2006, http://www.imc.org/ietf-openpgp/ mail-archive/msg14307.html.

[GCM] Dworkin, M., NIST Special Publication 800-38D, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", November 2007.

[IKEALG] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, December 2005.

[KEYSIZ] Orman, H. and P. Hoffman, “Determining Strengths For Public Keys Used For Exchanging Symmetric Keys”, BCP 86, RFC 3766, April 2004.

[KPR03] Klima, V., Pokorny, O., Rosa, T., “Attacking RSA-based Sessions in SSL/TLS”, http://eprint.iacr.org/2003/052/, March 2003.

[MODP] Kivinen, T. and M. Kojo, “More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)”, RFC 3526, May 2003.

[PKCS6] RSA Laboratories, “PKCS #6: RSA Extended Certificate Syntax Standard”, version 1.5, November 1993.

[PKCS7] RSA Laboratories, “PKCS #7: RSA Cryptographic Message Syntax Standard”, version 1.5, November 1993.

[RANDOM] Eastlake, D., 3rd, Schiller, J., and S. Crocker, “Randomness Requirements for Security”, BCP 106, RFC 4086, June 2005.

[RFC3749] Hollenbeck, S., “Transport Layer Security Protocol Compression Methods”, RFC 3749, May 2004.

[RFC4366] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, “Transport Layer Security (TLS) Extensions”, RFC 4366, April 2006.

[RSA] R. Rivest, A. Shamir, and L. M. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communications of the ACM, v. 21, n. 2, Feb 1978, pp. 120-126.

[SEQNUM] Bellovin, S., “Defending Against Sequence Number Attacks”, RFC 1948, May 1996.

[SSL2] Hickman, Kipp, “The SSL Protocol”, Netscape Communications Corp., Feb 9, 1995.

[SSL3] A. Freier, P. Karlton, and P. Kocher, “The SSL 3.0 Protocol”, Netscape Communications Corp., Nov 18, 1996.

[SUBGROUP] Zuccherato, R., “Methods for Avoiding the”Small-Subgroup” Attacks on the Diffie-Hellman Key Agreement Method for S/MIME”, RFC 2785, March 2000.

[TCP] Postel, J., “Transmission Control Protocol”, STD 7, RFC 793, September 1981.

[TIMING] Boneh, D., Brumley, D., “Remote timing attacks are practical”, USENIX Security Symposium 2003.

[TLSAES] Chown, P., “Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)”, RFC 3268, June 2002.

[TLSECC] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, “Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)”, RFC 4492, May 2006.

[TLSEXT] Eastlake, D., 3rd, “Transport Layer Security (TLS) Extensions: Extension Definitions”, Work in Progress, February 2008.

[TLSPGP] Mavrogiannopoulos, N., “Using OpenPGP Keys for Transport Layer Security (TLS) Authentication”, RFC 5081, November 2007.

[TLSPSK] Eronen, P., Ed., and H. Tschofenig, Ed., “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)”, RFC 4279, December 2005.

[TLS1.0] Dierks, T. and C. Allen, “The TLS Protocol Version 1.0”, RFC 2246, January 1999.

[TLS1.1] Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.1”, RFC 4346, April 2006.

[X501] ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models, 1993.

[XDR] Eisler, M., Ed., “XDR: External Data Representation Standard”, STD 67, RFC 4506, May 2006.

Working Group Information

The discussion list for the IETF TLS working group is located at the e-mail address tls@ietf.org. Information on the group and information on how to subscribe to the list is at https://www1.ietf.org/mailman/listinfo/tls

Archives of the list can be found at: http://www.ietf.org/mail-archive/web/tls/current/index.html

Contributors

Christopher Allen (co-editor of TLS 1.0) Alacrity Ventures ChristopherA@AlacrityManagement.com

Martin Abadi University of California, Santa Cruz abadi@cs.ucsc.edu

Steven M. Bellovin Columbia University smb@cs.columbia.edu

Simon Blake-Wilson BCI sblakewilson@bcisse.com

Ran Canetti IBM canetti@watson.ibm.com

Pete Chown Skygate Technology Ltd pc@skygate.co.uk

Taher Elgamal taher@securify.com Securify

Pasi Eronen pasi.eronen@nokia.com Nokia

Anil Gangolli anil@busybuddha.org

Kipp Hickman

Alfred Hoenes

David Hopwood Independent Consultant david.hopwood@blueyonder.co.uk

Phil Karlton (co-author of SSLv3)

Paul Kocher (co-author of SSLv3) Cryptography Research paul@cryptography.com

Hugo Krawczyk IBM hugo@ee.technion.ac.il

Jan Mikkelsen Transactionware janm@transactionware.com

Magnus Nystrom RSA Security magnus@rsasecurity.com

Robert Relyea Netscape Communications relyea@netscape.com

Jim Roskind Netscape Communications jar@netscape.com

Michael Sabin

Dan Simon Microsoft, Inc. dansimon@microsoft.com

Tom Weinstein

Tim Wright Vodafone timothy.wright@vodafone.com

Editors' Addresses

Tim Dierks Independent EMail: tim@dierks.org

Eric Rescorla RTFM, Inc. EMail: ekr@rtfm.com