ATTACKS AGAINST THE WAP WTLS PROTOCOL

Markku-Juhani Saarinen
University of Jyväskylä
P.O. Box 35
FIN-40351 Jyväskylä, Finland
mjos@jyu.fi

Abstract

The WAP WTLS protocol was designed to provide privacy, data integrity, and authentication for wireless terminals. The protocol is currently being fielded, and it is expected that the protocol will be contained in millions of devices in a few years.

Even though the WTLS protocol is closely modeled after the well-studied TLS protocol, we have identified a number of potential security problems in it. In this note, we describe a chosen plaintext data recovery attack, a datagram truncation attack, a message forgery attack, and a key-search shortcut for some exportable keys.

1. INTRODUCTION

The WTLS [18] (Wireless Transport Layer Security) protocol is the security layer of the WAP (Wireless Application Protocol). It is becoming the de facto standard for providing privacy, data integrity, and authentication for applications in cellular phones and other small wireless terminals. Millions of devices using WTLS are expected to be fielded worldwide before the end of the year 2000.

WTLS bears a close resemblance to the SSL [8, 19] and TLS [7] protocols, but a number of changes has been made to the protocol by the WAP Forum. These changes were motivated by the special requirements of the WTLS protocol:

- Both datagram and connection oriented transport layer protocols must be supported.
- The protocol must be able to cope with long round-trip times.

- The bandwidth of some bearers can be very low.
- The processing power of many mobile terminals is quite limited.
- The memory capacity of many mobile terminals is very modest.
- The restrictions on exporting and using cryptography must be considered.

In other words, the authors of WTLS took TLS and tried to add datagram support, optimize the packet size, and select fast algorithms into the algorithm suite.

2. SECURITY

At the surface, the WTLS looks reasonably good. Most of the text in the WTLS specification has been adopted, word to word, from the TLS specification. However, many of the changes that were made by WAP Forum have led to security problems.

Predictable IVs lead to chosen-plaintext attacks against low-entropy secrets. While the TLS protocol was designed to be used over a reliable transport (such as TCP/IP), the WTLS protocol should be able to operate over an unreliable datagram transport where datagrams may be lost, duplicated, or reordered. If CBC mode is being used, this requirement makes it necessary for the IV to be either contained in the packet itself (explicit IV, as in IPSec) or that the IV for that block is somehow derived from data already available to the recipient. WTLS uses a linear IV computation, even for reliable transports.

When a block cipher is used in CBC mode, the IV for encrypting each packet is computed as follows:

$$IV_s = IV_0 \oplus (s \mid s \mid s \mid s)$$

where s is a 16-bit sequence number of the packet and IV_0 is the original IV, derived during key generation. The plaintext blocks $P_{s,0}, P_{s,1}, \ldots$ in the packet s are encrypted as

$$\begin{aligned} C_{s,0} &= E_k(IV_s \oplus P_{s,0}) \\ C_{s,i} &= E_k(C_{s,i-1} \oplus P_{s,i}), for\ i > 0 \end{aligned}$$

Consider a terminal application (such as telnet), where each keypress is sent as an individual packet. Alice enters her password into this application, and Eve captures these packets. Eve now has blocks of type

$$C_{s,0} = E_k(P_{s,0} \oplus IV_0 \oplus (s \mid s \mid s \mid s))$$

where $P_{s,0}$ contains an unknown letter of Alice's password. Note that s is known to Eve.

Now somehow Eve gets hold of Alice's channel, perhaps through an echo feature in some application. Eve guesses that the unknown letter in the password is L. Eve sends the following packet through Alice's channel:

$$P_{r,0} = L \oplus (s \mid s \mid s \mid s) \oplus (r \mid r \mid r \mid r)$$

where r is the sequence number of this packet. One can see that because $(r \mid r \mid r \mid r)$ cancels out in the CBC computation, a right guess $L = P_{s,0}$ leads to matching ciphertexts $C_{r,0} = C_{s,0}$. In other words, this is an oracle that tells whether the guessed password letter was correct. The entire password can be brute forced, letter by letter, with a few hundred tests using this oracle.

While the above description of the attack is highly simplified, one can see that a too easily predictable IV leads to situations where low-entropy secrets can be read.

This attack is similar to the attacks described by Bellovin against the IPSec protocol in [3].

The XOR MAC and stream ciphers. The WTLS protocols supports, among other MACs, a 40-bit XOR MAC. The XOR MAC works by padding the message with zeros, dividing it into 5-byte blocks and xoring these blocks together. Note that this construction differs from the one presented in [2].

The specification states that the XOR MAC is only intended for "some devices with very limited CPU resources". The specification also tells us that that the XOR MAC "may not provide as strong message integrity protection as SHA" when export able encryption is being used. In fact it is easy to see that the XOR MAC does not provide any message integrity protection if stream ciphers are being used, regardless of the key length.

If one inverts a bit position n in the ciphertext, the MAC can be made to match by inverting the bit $(n \bmod 40)$ in the MAC. This can be repeated arbitrary number of times. Thus, when stream ciphers are used, the XOR MAC does not provide any integrity protection.

35-bit DES encryption. The 40-bit DES encryption method is defined to use five bytes of keying material. Because of the parity bits contained in each byte of a DES key, there are only $5 * 7 = 35$ effective key bits in five bytes. This amounts to a reduction of the keyspace by a factor of 32. We note that the 56-bit DES has the correct amount of keying material (8 bytes).

The protocol clearly does not meet its requirement of reaching the best possible security level in export-weakened encryption modes.

The PKCS #1 attack. The RSA signatures and encryption are performed according to PKCS # 1, version 1.5 [9]. Daniel Bleichenbacher and others have demonstrated that if the protocol includes an oracle that tells whether a given packet has a correct PKCS # 1 v 1.5 padding, RSA messages can be decrypted with approximately 2^{20} chosen ciphertext queries [6, 5]. In some implementations the WTLS error messages `bad_certificate` and `decode_error` may provide such an oracle to the attacker.

We recommend that the 2.0 version of the PCKS #1 should be used instead [10, 17].

Unauthenticated alert messages. Some of the alert messages used in the protocol are sent in cleartext and are not properly authenticated. Most of these messages are warnings and do not cause the the session to be terminated.

Since an alert message can take up a sequence number "slot" in the protocol, an active attacker may replace an encrypted datagram with an unauthenticated plaintext alert message with the same sequence number without being detected. This leads to a truncation attack that allows arbitrary packets to be removed from the data stream.

We recommend that all messages affecting the protocol state should be properly authenticated.

Other Plaintext leaks. Under exportable keys the initial IV of each packet can be determined by an eavesdropper from the Hello messages and the sequence number alone. We are not aware of export laws in any country that would mandate this.

The change of keys can be determined by an eavesdropper, because the `record_type` field is sent unencrypted. This field determines the type of the message; one type being the Change Cipher Spec type.

Also, the existence of encrypted error messages can be determined from the `record_type` field. The exact nature of the encrypted error messages can not be determined.

Probable plaintext attacks. In order to mount an exhaustive key search on a symmetric cipher, one needs to have enough known or probable plaintext, so that the correct key can be recognized with trial decryption of one or more blocks. Attacks of this type against the IPSec protocol have been considered in [4].

We have observed that brute force attacks against the block ciphers in WTLS can be easily mounted, because the correct keys can be always recognized with a trial decryption of the last block of each packet.

We assume that a 64-bit block cipher is used. The last block is padded to the next full 8-byte limit by filling it with the padding length. In other words, if the last byte of $E_k^{-1}(C_i) \oplus C_{i-1}$ is n, the preceding n bytes of the plaintext must also contain this number.

If this test is passed, the key can be furthermore verified with the last block of arbitrary number of packets.

A note on Diffie-Hellman key agreement. The WTLS specification includes 512- and 768-bit primes p_1 and p_2, along with generators, that are to be used in Diffie-Hellman computations. The group order of the multiplicative subgroup generated by the generator is not given.

The absence of the group order makes it impossible to check that the given public value belongs to the correct multiplicative subgroup, as in DSA and KEA [11, 13].

It is known that if the group order is relatively smooth, the discrete logarithm problem becomes substantially easier to an attacker that knows the the factorization of the group order [14].

To verify that the group order was not divisible with small factors of $p-1$ (for either of these groups), we ran Paul Zimmerman's GMP-ECM elliptic curve factoring program for a total of 100 hours of CPU time on 333 MHz Sun Ultra 10s.

We found 4 factors (largest 52 bits) of the 512-bit number $p_1 - 1$, leaving a 438-bit composite cofactor. We verified that the group order of group 1 is not divisible with the factors found.

We found 7 factors (largest 70 bits) of the 768-bit number $p_2 - 1$, leaving a 658-bit composite cofactor. We verified that the group order of group 2 is not divisible with the factors found.

We suspect that the group order information was left out from the specification, not because of an attempt to mount a back door into WTLS, but because the authors did not see the relevance of the group order to the security of Diffie-Hellman key exchange. The group orders of the elliptic curve groups are given. These are all prime.

3. CONCLUSIONS

We have identified a number of security flaws and shortcomings in the WAP WTLS protocol: a chosen plaintext data recovery attack, a datagram truncation attack, a message forgery attack, and a key-search shortcut for some exportable keys. WTLS clearly needs revision.

It is interesting to note that despite their close resemblance, the WTLS protocol appears to be more vulnerable to attacks than TLS. Most of the attacks presented in this paper against WTLS have previously appeared in research literature in some form. We stress the need for prudent security engineering, even when making minor changes to a communications security protocol.

4. ACKNOWLEDGEMENTS

This work is based on publicly available documents. The security problems described in this work differ from those that were found by the author in a preliminary evaluation of a confidential draft of the WAP WTLS protocol.

The author wishes to thank colleagues at SSH Communications Security and University of Jyväskylä for their encouragement, and the anonymous referees for their valuable comments.

References

[1] M. Bellare, R. Canetti and H. Krawczyk, "Keying Hash Functions for Message Authentication," *Advances in Cryptology – Crypto '96 Proceedings*, Springer-Verlag, 1996

[2] M. Bellare, R. Guérin and P. Rogaway, "XOR MACs: New Methods for Authentication Using Finite Pseudorandom Functions," *Advances in Cryptology – Crypto '95 Proceedings*, Springer-Verlag, 1995

[3] S. Bellovin, "Problem Areas for the IP Security Protocols," *Proceedings of the Sixth USENIX Security Symposium*, pp. 205–214, USENIX Association 1996

[4] S. M. Bellovin, "Probable Plaintext Cryptanalysis of the IP Security Protocols," *Proceedings of the Symposium on Network and Distributed System Security*, pp. 155 – 160, 1997

[5] D. Bleichenbacher, "Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1," *Advances in Cryptology – Crypto '98 Proceedings*, pp. 1 – 12, Springer-Verlag, 1998

[6] D. Bleichenbacher, B. Kaliski and J. Staddon, "Recent results on PKCS #1: RSA Encryption Standard," *RSA Laboratories' Bulletin*, Number 7, June 26, 1998.

[7] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," RFC 2246, `ftp://ftp.isi.edu/in-notes/rfc2246.txt`, 1999

[8] A. O. Freier, P. Karlton and P. C. Kocher, "The SSL Protocol Version 3.0," `http://www.netscape.com/eng/ssl3/draft302.txt`, 1996

[9] B. Kaliski, "PKCS #1: RSA Encryption Version 1.5," RFC 2313, `ftp://ftp.isi.edu/in-notes/rfc2313.txt`, 1998

[10] B. Kaliski and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0," RFC 2437, `ftp://ftp.isi.edu/in-notes/rfc2437.txt`, 1999

[11] National Institute of Standards and Technology, "Digital Signature Standard," FIPS PUB 186, 1994

[12] National Institute of Standards and Technology, "Secure Hash Standard," FIPS PUB 180-1, 1995

[13] National Security Agency, "Skipjack and KEA Algorithm Specifications Version 2.0," `http://csrc.nist.gov/encryption/skipjack-kea.htm`, 1998

[14] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance," IEEE Transactions on Information Theory, Vol. 24, pp. 106 – 110, 1978

[15] J. Pollard, "Monte Carlo Methods for Index Computation (mod p)," *Mathematics of Computation*, Vol 32., pp. 918 – 924, 1974

[16] R. Rivest, "The MD5 Message-Digest Algorithm," RFC1321, `ftp://ftp.isi.edu/in-notes/rfc1321.txt`, 1992

[17] M. Robshaw and J. Staddon, "A Note on the Security of the OAEP-Enhanced RSA Public-Key Encryption Scheme," *RSA Laboratories' Bulletin*, Number 9, February 23, 1999

[18] WAP Forum, "Wireless Application protocol – Wireless Transport Layer Security Specification, Version 12-Feb-1999," available from `http://www.wapforum.org`, 1999

[19] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 protocol," *Proceedings of the Second USENIX Workshop on Electronic Commerce*, USENIX Press, pp. 29–40, 1996