Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 16, 2012

N. Mavrogiannopoulos
KU Leuven - IBBT

May 15, 2012

Preventing cross-protocol attacks in TLS protocol
draft-mavrogiannopoulos-tls-server-key-exchage-00

Abstract

This memo proposes a fix in the TLS ServerKeyExchange message signature, to prevent cross-protocol attacks.

Status of This Memo

This Internet-Draft will expire on November 16, 2012.

Copyright Notice

Table of Contents

1. Introduction

The TLS protocol [RFC5246] suffers from an issue in the ServerKeyExchange message signature discovered by Wagner and Schneier in [WS-ATTACK]. They describe a cross-protocol attack on the SSL 3.0 [RFC6101] protocol, that re-uses a signed ServerKeyExchange packet in another session with a different key exchange algorithm. In effect the attack uses a server as an oracle that transmits signed ServerKeyExchange messages that can be relayed to another, unrelated, session. The described attack turned to be impossible to implement in practice, but the underlying idea is applicable to all TLS protocol versions, and as the supported key exchange algorithms increase, it provides a tool for new attacks on the protocol [ref needed].

In this document we propose a fix for the TLS protocol that does not require a protocol version upgrade.

2. Terminology

This document uses the same notation and terminology used in the TLS Protocol specification [RFC5246].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. The Wagner and Schneier attack

Wagner and Schneier in [WS-ATTACK] describe a cross-protocol attack (The authors refer to it as "key exchange algorithm rollback attack") based on the observation that the digital signature in a Diffie-Hellman key exchange does not cover any identifier of the negotiated ciphersuite. According to the SSL 3.0 protocol [RFC6101] when a Diffie-Hellman key exchange has been negotiated, the group parameters are sent by the server in the ServerKeyExchange message as shown below.

```
       struct {
           select (KeyExchangeAlgorithm) {
              case diffie_hellman:
                   opaque dh_p<1..2^16-1>;
                   opaque dh_g<1..2^16-1>;
                   opaque dh_Ys<1..2^16-1>;
                   Signature signed_params;
               case rsa:
                   opaque rsa_modulus<1..2^16-1>;
                   opaque rsa_exponent<1..2^16-1>;
                   Signature signed_params;
           };
       } ServerKeyExchange;
```

The signature on that message is calculated on the algorithm parameters, and the nonces exchanged by both peers. The crucial observation is that the negotiated key exchange algorithm is not part of this signature.

This omission allows an adversary to re-use a signed ServerKeyExchange packet in another session, with another key exchange algorithm, by initiating a parallel connection to the server. In particular, the described in [WS-ATTACK] attack deceives a client who advertises a TLS_RSA_EXPORT ciphersuite and expects temporary RSA parameters in the ServerKeyExchange message, into receiving Diffie-Hellman parameters from a TLS_DHE_RSA ciphersuite.

The attack is based on a wrong assumption in the TLS packet parsing, which prevents it from being practical. It demonstrates, however, the idea of a cross-protocol attack utilizing two of the SSL 3.0 key exchange methods, the Diffie-Hellman and the RSA-EXPORT key exchanges.

4. The new ServerKeyExchange signature

The goal of this memo is to restrict the applicability of the server provided signed ServerKeyExchange to the current session. A simple fix may be to include the negotiated ciphersuite into the signature. However, the TLS protocol is complex and a key exchange method does not always imply a single format of the ServerKeyExchange signature. For example, the elliptic curves key exchange method may be used with an arbitrary elliptic curve [RFC4492] which requires different data in the ServerKeyExchange than when used with a named curve. Such key exchange suboptions are negotiated using TLS extensions and such extensions should be covered by the signature, to prevent any attack that takes advantage of the different signature format.

For that we propose that the signature of the ServerKeyExchange

message to be modified to include in addition to explicit identifiers of the algorithms, all the previously exchanged messages. The proposed signature for a ServerKeyExchange message is shown below.

```
enum { server (0), client (1) } ConnectionEnd;

enum { dhe_dss (0), dhe_rsa (1),
       ec_diffie_hellman (2)
     } KeyExchangeAlgorithm;

struct {
  digitally-signed struct {
    ConnectionEnd entity;
    KeyExchangeAlgorithm kx_algorithm;
    select (KeyExchangeAlgorithm) {
      case dhe_dss:
      case dhe_rsa:
        ServerDHParams params;
      case ec_diffie_hellman:
        ServerECDHParams;
    }
    opaque handshake_messages<0..2^24-1>;
  }
} ServerKeyExchange;
```

The new format includes explicit indicators of the entity (server), the key exchange algorithm used, the handshake messages exchanged, and the parameters of the key exchange. This modification will be negotiated by using a new TLS extension to allow backwards compatibility.

5. The extension

In order for a client to advertise its support for the new ServerKeyExchange foramt we add a new extension "new_server_key_exchange", with value TBD-BY-IANA, to the enumerated ExtensionType defined in [RFC5246]. Clients MAY include the extension of type "new_server_key_exchange" in the (extended) client hello. The "extension_data" field of this extension MUST be empty.

Servers that receive an extended client hello containing a "new_server_key_exchange" extension, MAY accept the request for the new ServerKeyExchange format by including an extension of type "new_server_key_exchange" in the (extended) server hello. The "extension_data" field of this extension MUST be empty.

Note that this extension is valid for all versions of the TLS protocol including TLS 1.2 [RFC5246] and SSL 3.0 [RFC6101].

6. Security considerations

This extension modifies the ServerKeyExchange message in order to prevent attacks to the protocol similar in nature with the Wagner and Schneier attack. In order for the protection to be applicable, both the client and the server must support this extension. In the case where either or none support this extension, the old vulnerable format is being used.

7. IANA Considerations

This document defines the TLS extension "new_server_key_exchange" (value TBD-BY-IANA) whose value should be assigned from the TLS ExtensionType Registry defined in [RFC5246].

8. References

8.1. Normative References

[WS-ATTACK] Wagner, D. and B. Schneier, "WS-ATTACK of the SSL 3.0 protocol", The Second USENIX Workshop on Electronic Commerce Proceedings, USENIX Press , November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

8.2. Informative References

[RFC6101] Freier, A., Karlton, P., and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, August 2011.

[RFC4492] 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.

Author's Address

Nikos Mavrogiannopoulos
KU Leuven ESAT/SCD/COSIC - IBBT
Kasteelpark Arenberg 10, bus 2446
Leuven-Heverlee,   B-3001
Belgium

EMail: nikos.mavrogiannopoulos@esat.kuleuven.be