Network Working Group
Internet-Draft
Updates: RFCs 5246, 4366, 4347, 4346, 2246 (if approved)
Intended status: Standards Track
Expires: June 19, 2010

E. Rescorla
RTFM, Inc.
M. Ray
S. Dispensa
PhoneFactor
N. Oskov
Microsoft

December 16, 2009

Transport Layer Security (TLS) Renegotiation Indication Extension
draft-ietf-tls-renegotiation-02.txt

Abstract

SSL and TLS renegotiation are vulnerable to an attack in which the attacker forms a TLS connection with the target server, injects content of his choice, and then splices in a new TLS connection from a client. The server treats the client's initial TLS handshake as a renegotiation and thus believes that the initial data transmitted by the attacker is from the same entity as the subsequent client data. This specification defines a TLS extension to cryptographically tie renegotiations to the TLS connections they are being performed over, thus preventing this attack.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Conventions Used In This Document 4
3. Extension Definition 4
4. Renegotiation Protection Request Signalling Cipher Suite Value 6
5. Requirements for Sending and Receiving 6
6. Backward Compatibility 7
6.1. Client Considerations 7
6.2. Server Considerations 8
6.3. SSLv3 8
7. Security Considerations 8
8. IANA Considerations 9
9. Acknowledgements 9
10. References 10
10.1. Normative References 10
10.2. Informative References 10
Authors' Addresses 10

1. Introduction

TLS [RFC5246] allows either the client or the server to initiate renegotiation--a new handshake which establishes new cryptographic parameters. Unfortunately, although the new handshake is carried out using the cryptographic parameters established by the original handshake, there is no cryptographic connection between the two. This creates the opportunity for an attack in which the attacker who can intercept a client's transport layer connection can inject traffic of his own as a prefix to the client's interaction with the server. The attack [Ray09] proceeds as shown below:

```
Client                        Attacker                        Server
------                        -------                         ------
                                  <----------- Handshake ---------->
                                  <======= Initial Traffic ========>
<--------------------------  Handshake ============================>
<======================== Client Traffic ==========================>
```

To start the attack, the attacker forms a TLS connection to the server (perhaps in response to an initial intercepted connection from the client). He then sends any traffic of his choice to the server. This may involve multiple requests and responses at the application layer, or may simply be a partial application layer request intended to prefix the client's data. This traffic is shown with == to indicate it is encrypted. He then allows the client's TLS handshake to proceed with the server. The handshake is in the clear to the attacker but encrypted over the attacker's TLS connection to the server. Once the handshake has completed, the client communicates with the server over the newly established security parameters with the server. The attacker cannot read this traffic, but the server believes that the initial traffic to and from the attacker is the same as that to and from the client.

If certificate-based client authentication is used, the server will see a stream of bytes where the initial bytes are protected but unauthenticated by TLS and subsequent bytes are authenticated by TLS and bound to the client's certificate. In some protocols (notably HTTPS), no distinction is made between pre- and post-authentication stages and the bytes are handled uniformly, resulting in the server believing that the initial traffic corresponds to the authenticated client identity. Even without certificate-based authentication, a variety of attacks may be possible in which the attacker convinces the server to accept data from it as data from the client. For instance, if HTTPS [RFC2818] is in use with HTTP cookies [RFC2965] the attacker may be able to generate a request of his choice validated by the client's cookie.

Some protocols--such as IMAP or SMTP--have more explicit transitions between authenticated and unauthenticated phases and require that the protocol state machine be partly or fully reset at such transitions. If strictly followed, these rules may limit the effect of attacks. Unfortunately, there is no requirement for state machine resets at TLS renegotiation and thus there is still a potential window of vulnerability, for instance by prefixing a command which writes to an area visible by the attacker with a command by the client that includes his password, thus making the client's password visible to the attacker (note that this precise attack does not work with challenge-response authentication schemes but other attacks may be possible). Similar attacks are available with SMTP and in fact do not necessarily require the attacker to have an account on the target server.

It is important to note that in both cases these attacks are possible because the client sends unsolicited authentication information without requiring any specific data from the server over the TLS connection. Protocols which require a round trip to the server over TLS before the client sends sensitive information are likely to be less vulnerable.

These attacks can be prevented by cryptographically binding renegotiation handshakes to the enclosing TLS cryptographic parameters, thus allowing the server to differentiate renegotiation from initial negotiation, as well as preventing renegotiations from being spliced in between connections. An attempt by an attacker to inject himself as described above will result in a mismatch of the cryptographic binding and can thus be detected. The data used in the extension is similar to, but not the same as, the date used in the tls-unique and/or tls-unique-for-telnet channel bindings described in [I-D.altman-tls-channel-bindings], however this extension is not a general-purpose RFC 5056 [RFC5056] channel binding facility."

2. Conventions Used In This Document

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. Extension Definition

This document defines a new TLS extension: "renegotiation_info", which contains a cryptographic binding to the enclosing TLS connection (if any) for which the renegotiation is being performed. The "extension data" field of this extension contains a

"Renegotiation_Info" structure:

```
          struct {
            opaque renegotiated_connection<0..255>;
          } Renegotiation_Info;
```

The contents of this extension are specified as follows.

- o If this is the initial handshake for a connection, then the "renegotiated_connection" field is of zero length in both the ClientHello and the ServerHello. Thus, the entire encoding of the extension is: ff 01 00 01 00. The first two octets represent the extension type, the third and fourth octet the length of the extension itself, and the final octet the zero length byte for the "renegotiated_connection" field.
- o For ClientHellos which are renegotiating, this field contains the verify_data from the Finished message sent by the client on the immediately previous handshake. For current versions of TLS, this will be a 12-byte value.
- o For ServerHellos which are renegotiating, this field contains the concatenation of the verify_data values sent by the client and the server (in that order) on the immediately previous handshake. For current versions of TLS, this will be a 24-byte value.

The above rules apply even when TLS session resumption is used.

Upon receipt of the "renegotiation_info" extension, both client and server implementations which support the extension MUST verify that it contains the correct contents as specified above (the previous client Finished.verify_data in the ClientHello and the concatenation of both Finished.verify_data values in the ServerHello). If the contents are incorrect, then it MUST generate a fatal "handshake_failure" alert and terminate the connection. This allows two implementations both of which support the extension to safely renegotiate without fear of the above attack. This requirement to validate any received RI extension still applies even after previous handshakes on the same the connection or session did not negotiate the use of RI. Every handshake must be treated independently in this respect because the attacker may attempt to make an initial handshake appear as a renegotiation handshake or vice-versa.

This extension also can be used with Datagram TLS [RFC4347]. Although for editorial simplicity this document refers to TLS, all requirements in this document apply equally to DTLS.

4. Renegotiation Protection Request Signalling Cipher Suite Value

Both the SSLv3 and TLS 1.0/TLS1.1 specifications require implementations to ignore data following the ClientHello (i.e., extensions) if they do not understand it. However, some SSLv3 and TLS 1.0 implementations incorrectly fail the handshake in such case. This means that clients which offer "renegotiation_info" may find handshake failures. In order to enhance compatibility with such servers, this document defines a second signalling mechanism via a special signalling cipher suite value (SCSV) "TLS_RENEGO_PROTECTION_REQUEST", with code point 0xNN, 0xMM. This SCSV is not a true cipher suite and cannot be negotiated. It merely has exactly the same semantics as an empty "renegotiation_info" extension. Because servers ordinarily ignore unknown cipher suites, the SCSV can be added safely on any initial handshake, including SSLv2 backward compatibility handshakes.

Servers MUST treat receipt of TLS_RENEGO_PROTECTION_REQUEST exactly as if the client had sent an empty "renegotiation_info" extension and respond with their own "renegotiation_info" extension. This is an explicit exception to the RFC 5246 Section 7.4.1.4 prohibition on the server sending unsolicited extensions and is only allowed because the client is signaling its willingness to receive the extension via the the TLS_RENEGO_PROTECTION_REQUEST SCSV. TLS implementations MUST continue to comply with 7.4.1.4 for all other extensions. Servers MUST NOT select this SCSV in any handshake, as it does not correspond to any valid set of algorithms.

Because the SCSV is equivalent to an empty "renegotiation_info" extension, any ClientHello used for secure renegotiation MUST include the "renegotiation_info" extension and not the SCSV. [TODO: if WG consensus is that this applies to legacy negotiation, we will make this requirement more pointed. otherwise, we need an explicit exception for that case.]

Note that a minimal client which does not support renegotiation at all can simply use the SCSV in all initial handshakes. Any compliant server MUST generate a fatal "handshake_failure" alert and terminate the connection when it sees any (apparent) attempt at renegotiation by such a client. Clients which do support renegotiation MUST implement Section 3 as well.

5. Requirements for Sending and Receiving

TLS clients which support this specification MUST generate either the "renegotiation_info" extension or the TLS_RENEGO_PROTECTION_REQUEST SCSV with every ClientHello, including ClientHellos where session

resumption is being offered.

TLS servers MUST support both the "renegotiation_info" extension and the TLS_RENEGO_PROTECTION_REQUEST SCSV. TLS servers which support this specification MUST generate the "renegotiation_info" extension in the ServerHello in response to any client which offers either "renegotiation_info" or TLS_RENEGO_PROTECTION_REQUEST in the ClientHello. This includes ServerHellos which resume TLS sessions.

TLS servers implementing this specification MUST ignore any unknown extensions offered by the client and MUST accept version numbers higher than its highest version number and negotiate the highest common version. These two requirements reiterate preexisting requirements in RFC 5246 and are merely stated here in the interest of forward compatibility.

6. Backward Compatibility

Existing implementations which do not support this extension are widely deployed and in general must interoperate with newer implementations which do support it. This section describes considerations for backward compatible interoperation.

6.1. Client Considerations

If a client offers the "renegotiation_info" extension or the TLS_RENEGO_PROTECTION_REQUEST SCSV and the server does not reply with "renegotiation_info" in the ServerHello, then this indicates that the server does not support secure renegotiation. Because the above attack looks like a single handshake to the client, the client cannot determine whether the connection is under attack or not. Note, however, that merely because the server does not acknowledge the extension does not mean that it is vulnerable; it might choose to reject all renegotiations and simply not signal it. However, it is not possible for the client to determine purely via TLS mechanisms whether this is the case or not.

If clients wish to ensure that such attacks are impossible, they need to terminate the connection immediately upon failure to receive the extension without completing the handshake. Such clients MUST generate a fatal "handshake_failure" alert prior to terminating the connection. However, it is expected that many TLS servers that do not support renegotiation (and thus are not vulnerable) will not support this extension either, so in general, clients which implement this behavior will encounter interoperability problems. There is no set of client behavior which will guarantee security and achieve maximum interoperability during the transition period. Clients need

to choose one or the other preference when dealing with potentially unupgraded servers.

[TODO: this needs a discussion of what signal to send in legacy renegotiation to match the resolution of the paired TODO in Section 4.]

6.2. Server Considerations

If the client does not offer the "renegotiation_info" extension or the TLS_RENEGO_PROTECTION_REQUEST SCSV then this indicates that the client does not support secure renegotiation or is unwilling to use it. However, because the above attack looks like two handshakes to the server, the server can safely continue the connection as long as it does not allow the client to renegotiate. If servers wish to ensure that such attacks are impossible they need to refuse to renegotiate with clients which do not offer the "renegotiation_info" extension. Servers which implement this behavior MUST respond to such requests with a "no_renegotiation" alert [RFC 5246 requires this alert to be at the "warning" level.] Servers SHOULD follow this behavior.

In order to enable clients to probe, even servers which do not support renegotiation MUST implement the minimal version of the extension described in this document for initial handshakes, thus signalling that they have been upgraded.

6.3. SSLv3

While SSLv3 is not a protocol under IETF change control, it was the original basis for TLS and most TLS stacks are also SSLv3 stacks. The SCSV and extension defined in this document can also be used with SSLv3 with no changes except for the size of the verify_data values, which are are 36 bytes long each. TLS Clients which support SSLv3 and offer secure renegotiation (either via SCSV or "renegotiation_info") MUST accept the "renegotiation_info" extension from the server even if the server version is {0x03, 0x00} and behave as described in this specification. TLS Servers which support secure renegotation and support SSLv3 MUST accept SCSV or the "renegotiation_info" extension and respond as described in this specification even if the offered client version is {0x03, 0x00}.

7. Security Considerations

The extension described in this document prevents an attack on TLS. If this extension is not used, TLS renegotiation is subject to an attack in which the attacker can inject their own conversation with

the TLS server as a prefix of the client's conversation. This attack is invisible to the client and looks like an ordinary renegotiation to the server. The extension defined in this document allows renegotiation to be performed safely. Servers SHOULD NOT allow clients to renegotiate without using this extension. Many servers can mitigate this attack simply by refusing to renegotiate at all.

While this extension mitigates the man-in-the-middle attack described in the overview, it does not resolve all possible problems an application may face if it is unaware of renegotiation. It is possible that the authenticated identity of the server or client may change as a result of renegotiation.

[TODO: This section still needs to be adjusted to match the WG discussion.] By default, TLS implementations conforming to this document MUST verify that once the peer has been identified and authenticated within the TLS handshake, the identity does not change on subsequent renegotiations. For certificate based cipher suites, this means bitwise equality of the end-entity certificate. If the other end attempts to authenticate with a different identity, the renegotiation MUST fail. If the server_name extension is used, it MUST NOT change when doing renegotiation.

A TLS library MAY provide a means for the application to allow identity and/or server_name changes across renegotiations, in which case the application is responsible for tracking the identity associated with data it is processing. This may require additional API facilities in the TLS library.

8. IANA Considerations

IANA [shall add/has added] the extension code point XXX [We request 0xff01, which has been used for prototype implementations] for the "renegotiation_info" extension to the TLS ExtensionType values registry.

IANA [shall add/has added] TLS cipher suite number 0xNN,0xMM [We request 0x00, 0xff] with name TLS_RENEGO_PROTECTION_REQUEST to the TLS Cipher Suite registry.

9. Acknowledgements

This vulnerability was originally discovered by Marsh Ray. The general concept behind the extension described here was independently invented by Steve Dispensa, Nasko Oskov, and Eric Rescorla with refinements from Nelson Bolyard, Pasi Eronen, Michael D'Errico,

Stephen Farrell, Michael Gray, David-Sarah Hopwood, Ben Laurie, Bodo Moeller, Martin Rex (who defined TLS_RENEGO_PROTECTION_REQUEST), Peter Robinson, Jesse Walker, Nico Williams and other members of the the Project Mogul team and the TLS WG. [Note: if you think your name should be here, please speak up.]

10. References

10.1. Normative References

[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.

10.2. Informative References

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.

[I-D.altman-tls-channel-bindings]
Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", draft-altman-tls-channel-bindings-07 (work in progress), October 2009.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC2965] Kristol, D. and L. Montulli, "HTTP State Management Mechanism", RFC 2965, October 2000.

[Ray09]   Ray, M., "Authentication Gap in TLS Renegotiation".

Authors' Addresses

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Email: ekr@rtfm.com

Marsh Ray
PhoneFactor
7301 W 129th Street
Overland Park, KS 66213
USA

Email: marsh@extendedsubset.com

Steve Dispensa
PhoneFactor
7301 W 129th Street
Overland Park, KS 66213
USA

Email: dispensa@phonefactor.com

Nasko Oskov
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Email: nasko.oskov@microsoft.com